INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Spike based neural codes : towards a novel bio-inspired still image coding schema

Khaled Masmoudi — Marc Antonini — Pierre Kornprobst

N° 7302

May 2010

Thème BIO

ISSN 0249-6399 ISRN INRIA/RR--7302--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Spike based neural codes : towards a novel bio-inspired still image coding schema

Khaled Masmoudi , Marc Antonini* , Pierre Kornprobst †

Thème BIO — Systèmes biologiques
Équipes-Projets NeuroMathComp & UNSA–CNRS–I3S

Rapport de recherche n° 7302 — May 2010 — 45 pages

Abstract: We asked whether rank order coding could be used to define an efficient compression scheme for still images. The main hypothesis underlying this work is that the mammalians retina generates a compressed neural code for the visual stimuli. The main novelty of our approach is to show how this neural code can be exploited in the context of image compression. Our coding scheme is a combination of a simplified spiking retina model and well known data compression techniques and consists in three main stages. The first stage is the bio-inspired retina model proposed by Thorpe et al. This model transforms of a stimulus into a wave of electrical impulses called spikes. The major property of this retina model is that spikes are ordered in time as a function of the cells activation: this yields the so-called rank order code (ROC). ROC states that the first wave of spikes give a good estimate of the input signal. In the second stage, we show how this wave of spikes can be expressed using a 4-ary dictionary alphabet: the stack run coding. The third stage consists in applying, to the stack run code, a arithmetic coder of the first order. We then compare our results to the JPEG standards and we show that our model offers similar rate/quality trade-off until 0.07 *bpp*, for a lower computational cost. In addition, our model offers interesting properties of scalability and of robustness to noise.

Key-words: Bio-inspired coding, spiking retina, stack run, still image compression

* I3S
† INRIA, EPI NeuroMathComp

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Spike based neural codes : towards a novel bio-inspired still image coding schema

Résumé : Nous nous proposons dans ce travail de mettre en oeuvre un codeur dimages statiques inspire des strategies de codage dans le syst‘eme visuel chez les primates. La comprehension de ces schemas de representation des stimuli visuels a motive plusieurs etudes durant ces derni‘eres decennies. Les observations neurophysiologiques tendent ‘a confirmer la capacite de notre cortex ‘a acquerir, transmettre puis traiter les donnees presentes dans la sc‘ene, dans un delai fort reduit. 100 ms suffisent ‘a un primate pour analyser une sc‘ene naturelle complexe. Ces constatations laissent supposer que notre syst‘eme dacquisition, la retine, code de mani‘ere efficace le stimulus auquel nous la soumettons. Ce code serait dautant plus efficace quil est soumis ‘a des contraintes physiologiques. En particulier, la bande passante du canal de transmission, le nerf optique, est reduite. De plus, nous ne disposons que de 1 million de fibres nerveuses pour vehiculer le stimulus recueilli par 100 millions de cellules photo-receptrices. Dans la perspective de concevoir un moteur de compression dimages statiques bio-inspire, nous mettons en oeuvre un syst‘eme de codage in- troduit dans les travaux de Thorpe et al; puis nous etudions ses performances en terme de cout du code. Nous proposons ensuite des ameliorations possibles de ce mod‘ele, nous decrirons ensuite un nouveau formalisme permettant de mieux apprehender le probl‘eme en vu de travaux futurs, enfin nous exposons quelques pistes de reflexion pour son extension ‘a la video. La comprehension des principes aux fondements du mod‘ele de Thorpe, objet principal de notre etude, requiert une revue des travaux menes precedemment. Nous nous interessons dans ce chapitre aux etudes concernant le code neuronal de la retine. Ces travaux, bien que nayant pas pour finalite la compression dimages, ont permis de mettre en exergue les principaux mecanismes utilises au niveau du chemin de donnees visuel humain.

Mots-clés : codage Bio-inspire, retine spikante, stack run, compression d’image statique

Introduction

During the past two decades, research in still image compression originated several coding algorithms, especially the JPEG effort which lead to the International Standard (IS) of the same name. Since then, subsequent efforts followed the same scheme for conceiving lossy image coders. Namely, three main stages are considered for the source coding. First, we apply a forward transform to the image, second, we quantize the transformed data, and finally, we pass it through an entropic coder (see Figure 1).

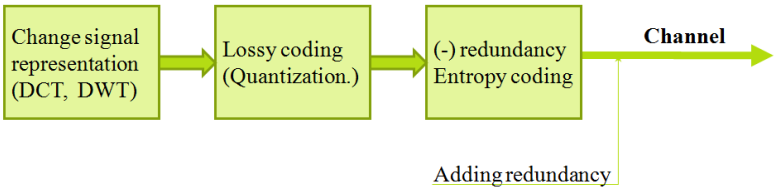


Figure 1: Typical block diagram for still image lossy coders as JPEG standards.

One of the main properties of such a scheme is to allocate the major part of the bandwidth available to the low frequency sub-bands. This feature is highly noticeable, especially when dealing with bandwidth restrictions. This was further enhanced in JPEG 2000 as this algorithm enables progressive decoding: low frequency sub-bands are transmitted (and obviously decoded) first, while higher frequency sub-bands are transmitted later in time. Such a design implies that the quality of the image being reconstructed gets higher as the bitstream is decoded.

Interestingly, we retrieve a similar behavior in the human retina. Indeed, we know that the neural code of the retina conveys low frequency sub-bands of the stimulus first, then higher frequencies as time goes [Woh08]. So, we investigated the analogy between the human visual system, and a JPEG-like coder. In our approach, the image stimulus is the data to encode, the retina is the image coder, and the optic nerve is the transmission channel toward the visual cortical areas. In addition, we know that this channel is noisy and has a restrained bandwidth. Thus, we naturally raised the question on how the retina represents the visual stimuli and transmits it in order to deal with the evoked constraints.

To answer this question, we first present, in Section 1, how the retina activity conveys the neural code triggered by the visual stimuli. We then overview, in Section 2, some state of the art studies interested in reproducing, interpreting, and compressing such a neural code. In Section 3, we focus on a simplistic retina simulator introduced in [Tho90, TFM96, RT01, RT02] aimed at reproducing some features of the biological retinas. We then specify, in Section 4, a complete coder/decoder scheme which we designed for compressing the output of the retina simulator. Finally, in Section 5, we compare our coder/decoder performance to already existing standards.

1 The neural code and spike-based coding schemes

The visual stimuli, presented to the human visual system, trigger a neural activity at the output of the retina. Concretely, the retina generates a set of electrical impulses termed as action potentials or also *spikes*. The spikes are emitted by a specific type of neurons tiling the surface of the retina, referred to as ganglion cells. A spike emitted by a ganglion cell is an all-or-none event, this means that it occurs fully or do not occur at all. Given a visual stimulus onset, each ganglion cell emits a series of spikes: the *spike train*. The set of these spike trains, defined over the set of ganglion cells, is the neural code of the visual stimuli. Note that all spikes have almost the same characteristic shape and amplitude, and thus spike amplitudes are independent of the stimulus intensity. This invariance of shape and amplitude of the spikes yielded a schematic representation of the neural code termed as the *raster plot*. The so-called raster plot is a binary-like array of values (see Figure 2). Deciphering this code is still a challenging issue in the field neurosciences.

The neural code of the retina is not a simple intensity signal transform. Indeed, several neurophysiologic experiments have shown that the neural code corresponding to a given stimulus encompasses many characteristics that could encode the information that is the most relevant to decode the stimulus [TFM96, RWdRvSB97]. Since [Adr26] it was claimed that the relevant information would be contained mostly in the mean spike firing rate. Whereas, in [PB68], authors suggested that information conveyed by each single spike could is important so that the nervous system can decode the visual stimulus. Since then, several coding models have been proposed to understand how could the visual information be *encapsulated* in the spike trains, and later interpreted by the nervous system. Figure 2 outlines some of the metrics that could be proposed in this perspective.

Basically, two sorts of coding schemes are proposed in the literature, rate codes and timing codes.

Rate codes: assume that spike firing rates convey most of the relevant information about the visual stimuli. We draw the reader attention to the fact that, rate codes, confusingly, refer to three distinct coding procedures: averaging over several experimental trials on a single neuron, averaging over neurons population in a defined time bin, and averaging spike count over a time window for a single neuron. The latter time averaging assumption (Figure 2 green lines) is the one that is commonly admitted in the literature. For a more detailed discussion of rate codes, see [RWdRvSB97, WW02]. Though, several studies such as [GT98], have shown that rate codes ability to discriminate visual stimuli is poor in many cases.

Timing codes: have been proposed to overcome some of the issues encountered with rate codes. A 'time-to-first-spike' strategy can be applied. For instance, in [Tho90, TFM96, RT01, RT02], authors consider only the first wave of spikes to be relevant with regard to a stimulus shown at time t_0 (Figure 2 red lines). This hypothesis relies on the experimental evidence that the more a neuron is stimulated the quicker it fires. A plausible neural code could be generated by ganglion cells firing asynchronously, where only the order, by which

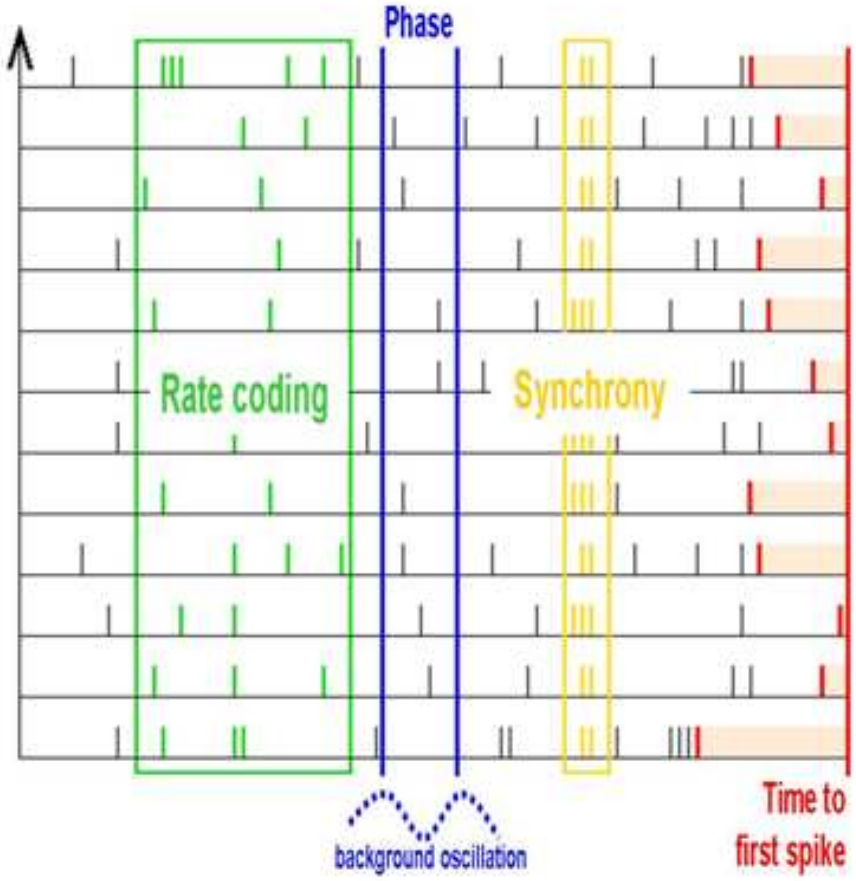


Figure 2: Some examples of possible relevant measures in the spiking code. The vertical axis represents a range of ganglion cells. The horizontal axis is oriented from right to left and represents time. Each line in the Figure is a spike train. *In green* Rate coding : For each cell, the firing rate is the spike count over a predetermined time window, of width T, divided by T. *In blue* Phase coding : the phase of spike signals is measured with respect to a background oscillation (here in dotted line). *In orange* Synchrony coding: ganglion cells reacting to a stimulus region "belonging the same object" fire simultaneously. Cells are thus grouped into classes. This defines a classification scheme used in bio-inspired segmentation. *In red* Time to first spike coding: The code output is the time to emit a first spike for each ganglion cell. In contradiction with the previous coding scheme, this assumes complete asynchrony in the encoding procedure.

these cells emit their first spike, is stimulus specific: this delineates the rank order coding. We will describe with further details a coder based on this assumption, later in this work. As an alternative, we can consider an analogous strategy, where the measure is not the time of spike firing but its phase. Such an approach consider that a stimulus alters a predefined background oscillatory signal. Neural output could then encode information in the phase of spikes with respect to this background oscillation (Figure 2 blue lines). Another strategy is to group neurons into classes which elements fire simultaneously. This coding scheme is mainly used for segmentation purposes. The underlying idea is that simultaneous firing could mean "belonging to the same object region" in the stimulus area (Figure 2 orange lines).

Besides these classical hypothesis, several ideas emerged. For instance, authors in [GM08] report that, at the level of the retina, ganglion cells encode the spatial structure of a visual stimulus in the relative timing of their first spikes. Also at the level of LGN, a subsequent processing stage in the visual system, the authors in [LS04] showed that the first spike bursts signal novelty to the visual cortex, while subsequent spikes add precision to the stimulus perception. Another example is presented in [BMLF], where authors showed the activity neurons in V1, an area of the visual cortex, to be depending on image statistics. Indeed experiments showed this activity to be more predictable when the retina is subject to natural stimuli, in comparison to artificial images. Readers interested in a more complete overview of possible coding strategies in the retina may refer to [RT02] or [WW02] for a detailed discussion.

Example strategies above, though not exhaustive, show spike-based coding strategies to be numerous in the literature. As our basic issue is to use one of these spiking codes for a static image coding scheme, we naturally raised the question on how are spikes generated in the retina. If the retina is a system that transforms a continuous input visual signal into a series of spikes, then how could it be implemented. Several models have been proposed to reproduce this transform, some of which are described in Section 2.

2 Generating, cracking, and compressing the neural code : An overview

In order to devise bio-inspired coding schemes we need to study stimuli representation as generated by the retina. First, we overview how, in the literature, simulators reproduce such a code (Section 2.1), the way decoders interpret it (Section 2.2), and the way encoders compress it (Section 2.3).

2.1 Generating a retinal code

In this Section we overview some retinal models of the literature, aimed at reproducing behavioral features found in mammalian's retina. We first present models mimicking, with high fidelity, retina functional stages. These are aimed at reproducing biologically realistic output R given some input stimulus I (Section 2.1.1). We then describe bio-inspired image transform models, which integrate biological features using existing image processing algorithms (Sec-

tion 2.1.2). Although the output of these models is not always a spiking code, they help understand mechanisms underlying the retina functioning.

2.1.1 Toward a biologically realistic model

Three retinal models are presented in this Section. Each one describing a specific stage in the commonly used architecture for retina simulators. These are sorted in chronological fashion from models restricted to a linear filtering stage (Section 2.1.1), to models adding a non linear gain control (Section 2.1.1), ending with retina simulators enabling spike generation with precise timings (Section 2.1.1).

A linear retinal model First attempts to devise retinal models considered a one-stage of linear filtering. For example, in [Rod65], authors describe a retinal model based on this architecture. The input of it is a continuous functional $I(x,t)$ over space x and time t representing the light intensity distribution. The output is measured, for every single ganglion cell, as the instantaneous spikes firing rate $R(t)$. Authors focused on a particular type of cells in the retina, namely the ganglion cells. Ganglion cells tile the deepest layer of the retina and are directly connected to the optic nerve. According to their polarity, these cells are classified into On-center or Off-center cells. An On (resp. Off) -center cell responds only at the onset (resp. offset) of the stimulus $I(x,t)$. Recordings made at the output of ganglion cells showed their behavior to be well approximated by linear filters both in space and time [Rod65]. First, spatial behavior of the ganglion cell is similar to a difference of Gaussians (*dog*) filtering. Keeping the same notations as in in [MI99], the spatial filter can be expressed as follows :

$$A(x) = K_c e^{-\frac{x^2}{2{r_c}^2}} - K_s e^{-\frac{x^2}{2{r_s}^2}}, \tag{1}$$

where K_c, K_s, r_c, and r_s are constant parameters. Second, neurophysiologic experiments also showed the output of the neurons to be subject to a temporal undershoot. A temporal filter allowing a good approximation of this can be expressed as follows :

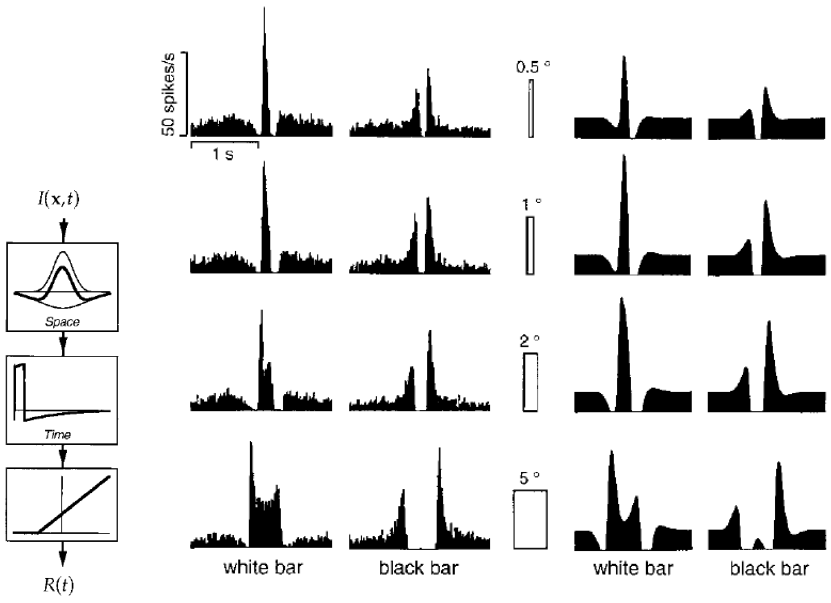
$$B(t) = \delta(t) - he^{-\frac{t}{\tau}}. \tag{2}$$

Figure 3(a) shows the profiles of these two filters.

For implementation convenience, authors made the assumption of space/time separability of the filtering process, thus yielding a model filter $H(x,t) = A(x).B(t)$. In reality this assumption is not rigorously met. Having this definition of H, authors made an estimate of $R(t)$ by integrating the input I weighted by H filter, over space and until time t. The result is then corrected, because ganglion cells may fire with no stimulus excitation. Typically a baseline frequency R_0 is added to model this spontaneous activity. At last we have:

$$R(t) = R_0 + \int_x \int_{t'} I(x,t')A(x)B(t-t')dxdt', \tag{3}$$

where $R(t)$ is expressed in spikes per second ($spike\ s^{-1}$). To validate this approach, authors recorded in vivo several responses to simple stimuli (spots, translating bars...). This allows the estimate of $R(t)$. These actual measurements are plotted and compared to model predictions as shown in Figure 3(b).



(a) (b)

Figure 3: Rodieck model outline schema. 3(a) From top to bottom : the spatial filter $A(x)$, the temporal one $B(t)$, and the integral of the spontaneous firing rate R_0. 3(b) Prediction power of the retina model output $R(t)$ (on the right) compared to actual data (on the left) (from [Rod65]).

Authors remarked that the separable filter used allows a good estimate of the firing rate $R(t)$ of a single neuron. The estimate is made more accurate when authors tested a modified *dog* filter. A temporal delay between central Gaussian of the *dog* filter and the surrounding one was implemented. This reproduces the actual behavior of a biological retina. Thus a bio-plausible retinal code, upto the assumptions made, can be generated using Rodieck retina model.

Although, if more sophisticated stimuli are presented, this model retina predictions fail matching actual retina output recordings. This is especially the case when light variations are high. In [Vic87], authors show the linearity of the filter to be valid only under strong constraints. A second model was then introduced in [Vic87], by adding a second non-linear stage as to reproduce more accurately the biological retina behavior.

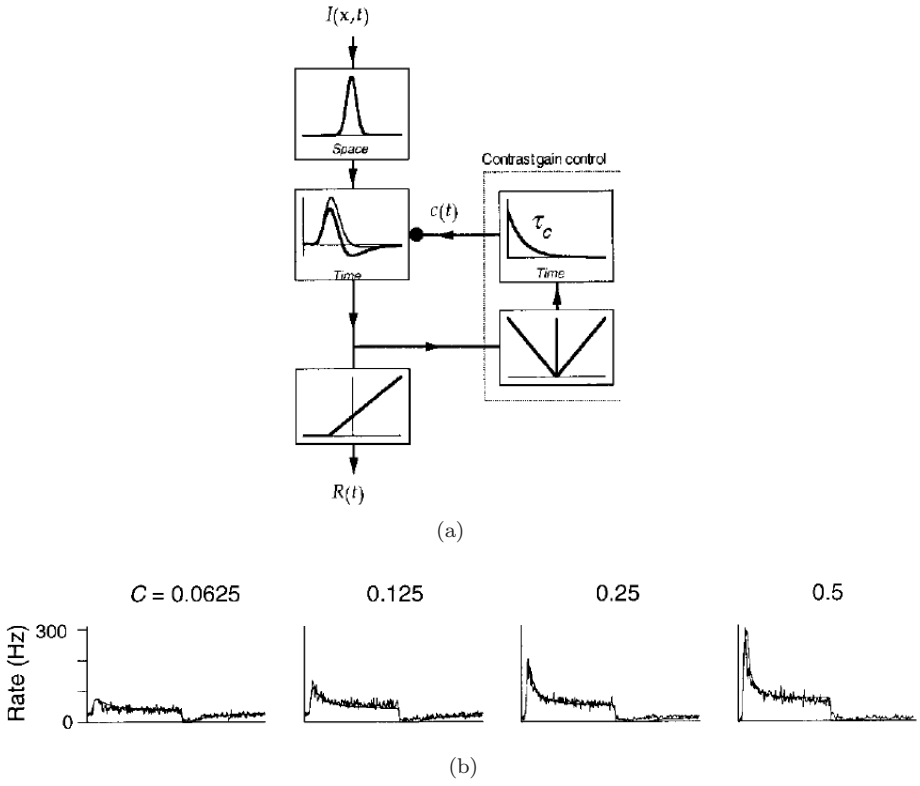
A linear/non-linear cascade model In [Vic87], the authors proposed a double-stage retina model cascading a linear then a non-linear filter. The models following this general schema are referred to as LNL models. As in the linear model described above, the input is light distribution $I(x,t)$, and the output is the instantaneous firing rate $R(t)$ of a given neuron. The first stage of the model is analogous to the linear filtering in [Rod65], i.e., a space/time separable process. Authors then add a highly non-linear rectification for gain control. Indeed, if large fluctuations occur in the input signal, the filter gain decreases and its waveform sharpens. This shows retina to be a stable encoder, as it is less sensitive, in terms of response amplitude, to high fluctuations of light though it responds faster. The non-linear gain control stage ensures this stability. For gain control implementing, the signal got after $I(x,t)$ is passed through the spatial/temporal filtering stage is rectified. This consists in a low pass time filtering with a time constant τ_c, followed by a full-wave rectifier which is a highly non-linear operator. A final linear averaging process is made to obtain an estimate of the output $R(t)$.

Actual measures of $R(t)$ are compared to the model predictions. Authors denote a good match between real and predicted measures. This model had some success modeling several classes of ganglion cells as Y-cells and P-cells. Nevertheless, stimuli responses that can be simulated with certain accuracy are also limited. In addition, basic assumptions of this model are not always true. The interdependency of neurones for spike generation has been shown in several studies, as well as the non-separability of time/space filtering in the retina.

Another limitation comes from restricting the output of the virtual retina to a simple firing rate calculus. This, obviously, occlude many aspects of the neural code. Thus models integrating other coding features of the retina allow a more realistic behavior. Such a model has been suggested in [WK09] as will be described in the next Section.

A linear-non linear cascade and a spike generator LNL models described in Section 2.1.1 add a gain control stage to simplistic linear filtering models described in Section 2.1.1. Still LNL models do not renders the output of a retina as it is conveyed to the visual cortex. A third stage was then added to the two already specified in LNL models, namely, a spike generator.

For instance, in [WK09], authors described a retina model integrating these three stages. The input is the visual stimulus $I(x,t)$, the output is the spiking



(a)

(b)

Figure 4: Victor model outline schema. In the first column of 4(a), from top to bottom, are shown the spatial, then the temporal filter and the spontaneous firing rate R_0 integral. The spatial linear filter is a *dog*-like filter comparable to [Rod65]. In the given case, coding processes of the first and second Gaussian are separate. Only the first Gaussian is shown here. The second column in 4(a) shows the low-pass temporal filter(top) and the full-wave rectifier(bottom) used for gain control. In 4(b) actual rate measures (jagged line) are compared to model prediction (smooth line). The model shows good estimates for $R(t)$(from [Vic87]).

signal. The exact timing of each spike is rendered. Besides, since all spikes are supposed to have the same shape, this yields a binary-like neural code as output. The model aims at reproducing the neural code with high fidelity to biological retina behavior.

To achieve this, three features are added to the retina model. In the first stage, the spatial/temporal filters of the model are non-separable. In the second stage, a new bio-plausible contrast gain control mechanism is introduced (see also [Woh07] for a mathematical study). And finally leaky integrate and fire neurons (nLIF's) are added to simulate the spike firing mechanism of the retina. The equations regulating the firing mechanism are the following :

$$\begin{cases} \dfrac{dV}{dt} = I_{gang}(x,t) - g^L V(t) + n_v(t) \\ \\ \text{fire a spike when threshold is reached at time } t_s : V(t_s) = 1 \\ \text{then refractory period : } V(t) = 0 \text{ while } t < t_s + n_r, \end{cases} \tag{4}$$

where V is a given neuron voltage, I_{gang} is the light distribution obtained after passing through linear filtering and gain control stages, g^L is a leaking conductance factor, and $\{n_v, n_r\}$ are additive noise signals.

Besides these biologically-realistic models, other attempts have been made to devise models that do not aim at mimicking exactly the physiological retina behavior. These take into account some characteristics already evoked, but use them to conceive spike coders/decoders in an image processing fashion. Some of these models, that will be referred to as bio-inspired image transforms, will be discussed in Section 2.1.2.

2.1.2 Bio-inspired image transforms

Several computer vision-based retina models are cited in the literature, offering variable complexity algorithms. As for biologically realistic models, first approaches restrict themselves to linear modeling of the retina, others integrate non linear stages. We will focus on two linear models in [OSL01] and [RT01], then we will discuss a non linear one specified in [LS07].

Linear models In [OSL01], authors present a wavelet-based retina model aimed at encoding a static image $I(x)$, and outputting a list of k coefficients $(c_i)_{0 \leqslant i \leqslant k-1}$ representing the neural code. Obviously this is not a realistic neural code. The model is based on a wavelet transform, i.e., the image is projected on a basis of vectors and is encoded by the series of projection coefficients $(c_i)_{0 \leqslant i \leqslant k-1}$. The issue encountered is to find the so-called wavelet basis. Formally the image is represented as a superposition of functions $(b_i)_{0 \leqslant i \leqslant k-1}$, scaled with $(c_i)_{0 \leqslant i \leqslant k-1}$ factors, thus yielding a linear model :

$$I(x) = \sum_i c_i b_i + \nu(x), \tag{5}$$

The (b_i)'s are a set of wavelet functions $((\psi_i)_{0 \leqslant i \leqslant k-2})$ and a scaling function ϕ (see [Mal89] for wavelet and scaling function definitions). (ψ_i) and ϕ functions are the model parameters to which authors add other parametrization features, the ensemble forms the parameters set θ. A learning process is implemented

to infer θ. The model is trained over a set of whitened natural images. The optimal $\hat{\theta}$ is the one that allows the minimum code length L for a given image I, also referred to as entropy (Entropy will be discussed in Section 5). This corresponds to minimizing the average log probability, of the random variable $(I|\theta)$. Once a solution $\hat{\theta}$ is obtained, the retina model is well defined. When applied to a visual stimulus $I(x)$, this model originates a sparse representation of the visual stimulus $I(x)$, using few coefficients $(c_i)_{0\leqslant i\leqslant k-1}$.

Another wavelet based algorithm is introduced in [Tho90, TFM96, RT01, RT02]. The input is a static image $I(x)$. Unlike the model in [OSL01] described above, the wavelet basis is known a priori. The mother wavelet waveform is as specified in (1). This kind of filtering maps the spatial behavior of a ganglion cell. The other basis vectors being deduced by scaling and translating this mother wavelet. After the wavelet coefficients $(c_i)_{0\leqslant i\leqslant k-1}$ are computed, they are sorted in the decreasing order of their absolute values. This yields a sorted series of coefficients $(c_i)_{0\leqslant i\leqslant k-1}$. The series of vectors corresponding to (c_i) coefficients is denoted by $(v_i)_{0\leqslant i\leqslant k-1}$. Authors then map $(v_i)_{0\leqslant i\leqslant k-1}$ into a series of indexes $(r_i)_{0\leqslant i\leqslant k-1}$, such that, r_i is the index in the basis of the vector v_i. In a way, basis vectors are sorted according to the decreasing order of the corresponding projection coefficients. The neurophysiologic experimental evidence underlying this processing is that, the more a ganglion cell is activated the quicker it fires. The spike-like code generated by this model is an order of firing of cells, i.e., $(r_i)_{0\leqslant i\leqslant k-1}$, and missing data is recovered through a priori known information. This is referred to as rank order coding. This algorithm is implemented in this work, as a basic brick for a coding scheme. The different processing levels will be discussed with further details in Section 3. Some enhanced versions of this algorithm are presented in the literature. For example, in [PST04], the authors use a matching pursuit procedure in order to eliminate redundancies between coefficients; and in [SF07], the authors proposed to invert the filters basis using a Moore-Penrose pseudo-inversion procedure for the same purpose.

Although attractive, these models do not render the nonlinearity observed in the retina. The model described next Section take into account this feature.

A linear/non linear cascade model Bio-inspired retina transforms evolved in the same fashion as biologically realistic models. Attempts have been made to conceive a double stage image transform with a linear/non linear cascade [LS07, VSN03]. The first stage is a linear filtering followed by a non-linear correction analogous to what is described in 2.1.1. Here we focus on the model introduced in [LS07].

Authors specify a linear/non linear invertible cascade image transform. The input of the model is an image $I(x)$. The linear filtering is a wavelet multi scale transform yielding a series of coefficients $(c_i)_{0\leqslant i\leqslant k-1}$ (cf. (5)). This transform is analogous to decompositions described in Section 2.1.2 (see [SF95] for further details).

The model second stage is a local non linear gain control feature, also known as *divisive normalization*. This stage aims at reducing dependencies between coefficients, thus avoiding the transmission of redundant information. This gain

control is biologically motivated by the mechanisms of lateral inhibition where highly activated cells inhibit neighbors. In an image processing application, reducing statiscal dependencies is often done using linear processes as principal components analysis (PCA), but high order dependencies remain, while divisive normalization reduces them. In order to specify divisive normalization, let us denote the probability density functions of wavelet coefficients by $(P(c_i))_{0\leqslant i\leqslant k-1}$. Authors then assume that :

$$P(c_i|(c_j)_{0\leqslant j\leqslant k-1, j\neq i}) = P(c_i|(c_j)_{j\in N(i)}), \tag{6}$$

where $N(i)$ is a neighborhood of c_i specified a priori. (6) means that the density of a given coefficient conditioned on all other coefficients equals the density conditioned on a local neighborhood : this yields the definition of a Markov field over wavelet coefficients space. Divisive normalization goes into two steps. The first step is to determine the Markov field parameters (w_j, d, a_j) defined in the following :

$$E(c_i|(c_j)_{j\in N(i)}) = \mu_i \equiv \sum_{j\in N(i)} w_j c_j \tag{7}$$

$$E((c_i-\mu_i)^2|(c_j)_{j\in N(i)}) = \sigma_i^2 \equiv d + \sum_{j\in N(i)} a_j (c_j-\mu_j)^2, \tag{8}$$

where μ_i and σ_i^2 are the conditional mean and variance of $c_i|(c_j)_{j\in N(i)}$. The second step is to correct coefficients $(c_i)_{0\leqslant i\leqslant k-1}$ to obtain transformed coefficients $(r_i)_{0\leqslant i\leqslant k-1}$ as follows :

$$r_i = \frac{c_i - \mu_i}{\sqrt{\sigma_i^2}} \tag{9}$$

This type of model can explain, to some extent, nonlinearities in the responses of mammalian ganglion cells (see [LS07] for a discussion of biological relevance of the method).

Now that we reviewed some methods to generate spike-like codes, a question that arises is how to do the inverse transform. Section 2.2 discusses this issue in the case where actual neural recordings are to be decoded, and in the case when coefficients of a bio-inspired model are considered.

2.2 Cracking the neural code

An important issue encountered in the study of spike-like codes is to decode them. We review some of the studies that have been made in this purpose. In Section 2.2.1, we present some methods for decoding the actual recorded neural data and, in Section 2.2.2, we present inversion algorithms of some bio-inspired transform models.

2.2.1 From actual cell recordings

Several studies investigated the possibility of visual stimuli reconstruction starting from actual neural data originated from the retina. In [BRdRvSW91, RWdRvSB97], authors make an attempt resolving this issue in a statistical fashion. The basic principle is to choose the more plausible visual signal input

among elements of a dictionary, knowing the output of the retina. In this problem, the known data is the retina output (t_i), representing the spike timings of a given neuron, and the unknown is $s(\tau)$, representing the waveform of the input stimulus. Formally, the reconstruction problem is mapped into the estimation of the density functional, $P(s(\tau)|(t_i))$. The input signal to reconstruct is the one that maximizes the probability $P(s(\tau)|(t_i))$. This is a maximum a posteriori (MAP) formulation of the problem. Details of density probability estimation method are given in [dRvSB88].

An alternative approach to (MAP) is to model decoding process using unknown filters, then infer these filters by learning over the set of all possible stimuli. A possible decoding algorithm is proposed and is expressed as follows :

$$\hat{s} = \sum_i F_1(t - t_i) + \sum_{i,j} F_2(t - t_i, t - t_j) + ... \tag{10}$$

where (F_n) are filters from the filters set $\mathbb{F}$, chosen to minimize a posteriori distortion of the reconstructed signal $\hat{s}(t)$, which is formally expressed as follows:

$$F_n = \underset{\mathbb{F}}{\operatorname{argmin}} \left(\int |s(t) - \hat{s}(t)|^2 dt \right). \tag{11}$$

Here the reconstruction problem is mapped into the estimation of (F_n) filters. Authors made experiments on H1 neurons in the fly visual system. These neurons are horizontal angular velocity detectors, thus input and output signal of the model are angular velocity trajectories. A comparison between the actual input signal and the reconstructed one is given in Figure 5. An analytic expression of such filters, if restriction is made to first order reconstruction, is given in [BRdRvSW91].

Another decoding approach is introduced in [SLD99]. The known data is the output of neurons in the lateral geniculate nucleus (LGN) which is a visual spiking code similar to the retinal code. The unknown is the visual stimulus represented in the LGN. A linear model of the LGN, analogous to linear models of the retina is used. Unlike [BRdRvSW91], authors do not consider each neuron output separately. Instead, ensembles of temporal stimuli/output are considered. The visual stimulus is denoted as $(s_0, ..., s_i, ..., s_n)$ where s_i is a time-dependent stimuli submitted to the i^{th} neuron. The output is binned and measured as instantaneous firing rates $(r_0, ..., r_i, ..., r_n)$, where r_i is the time dependent response of the i^{th} neuron.
The filters of the linear model are $2D + t$ filters, thus forming $3D$-tensors. For reconstruction to be done, the filters are inverted. This yields the following expression for h, the reverse filter matrix:

$$\begin{pmatrix} h_{11} & ... & h_{1n} \\ . & ... & . \\ . & ... & . \\ h_{1m} & ... & h_{mn} \end{pmatrix} = \begin{pmatrix} P_{r_1 r_1} & ... & P_{r_1 r_n} \\ . & ... & . \\ . & ... & . \\ P_{r_1 r_m} & ... & P_{r_m r_n} \end{pmatrix}^{-1} \begin{pmatrix} P_{r_1 s_1} & ... & P_{r_1 s_n} \\ . & ... & . \\ . & ... & . \\ h_{r_1 s_m} & ... & P_{r_m s_n} \end{pmatrix}. \tag{12}$$

Terms of the equation are sub-matrices. h_{ij} is the L-length time vector representing the reverse filter mapping the i^{th} neuron rate output to the j^{th} neuron input. $P_{r_i r_j}$ is the temporal cross correlation matrix, representing the correlation between the response of the i^{th} and j^{th} neurons. This ensures that

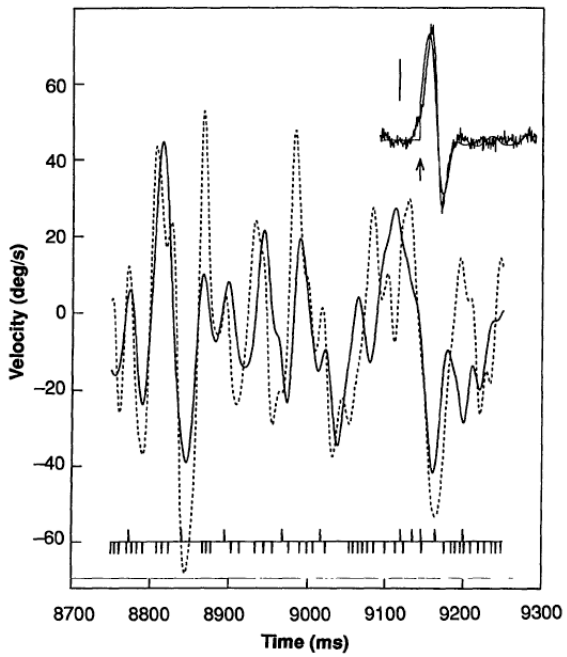


Figure 5: A reconstruction example of angular velocity trajectories from the output of H1 neurons in the fly visual system. In solid line : First order reconstruction. In dotted line : actual neural recordings. In the bottom : The spike train output corresponding to the angular velocity input (from [BRdRvSW91]).

correlations between all neurons are taken into consideration. $P_{r_i s_j}$ is a column vector specifying the relationship between the j^{th} neuron stimulus and the i^{th} neuron output. The reconstruction is according to this decoding model a sum of the convolution of rate responses with the reverse filters specified earlier. This leads to the following reconstruction process.

$$\hat{s_i}(t) = \sum_{j=1}^{n_r} \sum_{u=-L+1}^{u=L-1} h_{ij}(u)\, r_i(t-u) \tag{13}$$

Figure 6 shows an example of a movie reconstruction using this technique. Moving objects are fairly recognizable.

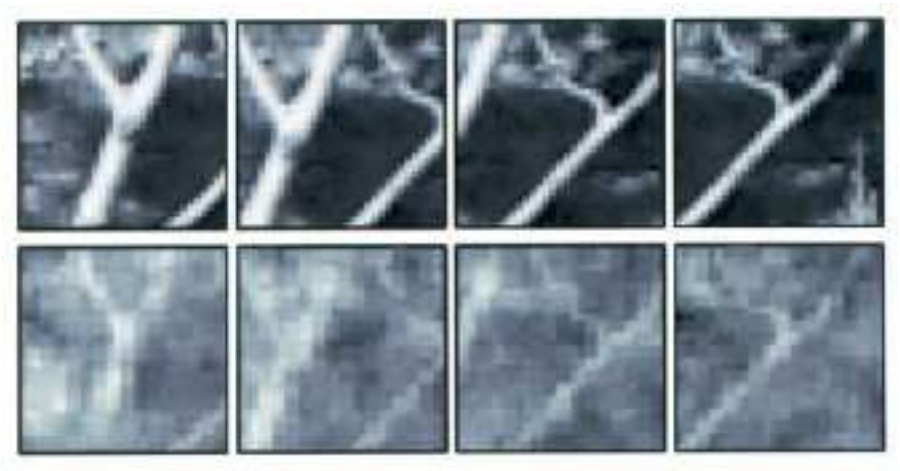


Figure 6: A reconstruction example of a movie from the output of LGN neurons in mammalians visual system [SLD99]).

The inverse transform shown in Figure 6 emphasizes the major limitation of actual neural data decoding techniques, which is their poor visual quality. Although objects are recognizable, these algorithms do not allow faithful image or video coding/decoding. Using bio-inspired model as described in 2.1.2 is an alternative to that limitation, as models are generally designed to be invertible.

2.2.2 From simulated retina models outputs

As discussed in 2.1.2, several bio-inspired image transforms have been implemented. In the algorithms presented the output of such models is a set of wavelet coefficients $(c_i)_{0 \leqslant i \leqslant k-1}$. The inversion of the transform could then be done by a simple sum of the basis vectors (b_i) weighted by the corresponding projection coefficients (c_i) as in [RT01, OSL01, SF95] yielding the following expession:

$$I(x) = \sum_i c_i b_i, \tag{14}$$

to which could be added a noise ν as in (5).

In the cases where the basis is not orthonormal, possible enhancements are proposed as to modify the coder or the decoder algorithm.

In [PST04], the coding is modified using a matching pursuit algorithm. This algorithm subtracts from a given coefficient c_i, corresponding to a vector b_i, the contribution of non orthogonal vectors $(b_j)_{j\neq i}$. This yields an exact reconstruction of the input $I(x)$.

In [SF07], the decoding procedure is modified. The dual basis $(b_i^*)_{0\leqslant i\leqslant k-1}$ of $(b_i)_{0\leqslant i\leqslant k-1}$ is used for the stimulus reconstruction. This yields the following expression for the inverse transform:

$$I(x) = \sum_i c_i b_i^*, \tag{15}$$

The dual basis is obtained by an Moore-Penrose inversion of the coding basis. This is generally done in two steps. First the vectors basis matrix B is SVD decomposed as to have:

$$B = U\Sigma V^*, \tag{16}$$

where U and V are unitary matrices and Σ a diagonal matrix. Second, the inversion of B to obtain dual basis vectors matrix B^* is then obtained by computing:

$$B^* = V\Sigma^* U^*, \tag{17}$$

In [WKA09], an inversion method for the model in [WK09] is proposed by modifying both in the coder and decoder level. The input to recover is continuous visual stimulus $I(x,t)$. At the coder level the (dog) spatial filter is replaced by a Dirac minus Gaussian filter defined by:

$$A(x) = K_c\delta_0 - K_s e^{-\frac{x^2}{2{r_s}^2}}, \tag{18}$$

where K_c, K_s, and r_s are constant parameters. A temporal filtering is added to form separable space/time filtering stage. Obviously, the model looses some of its biologically realistic features. This is done for convenience reasons. Indeed, at the decoder level, authors use a Moore-Penrose pseudo-inverted basis for reconstruction. The inversion is not stable unless these modifications are made. The inverse transform yields faithful though not exact reconstruction.

As shown in this Section, attempts have been made for transforming then inverse transforming visual inputs, in a bio-inspired fashion. Having these coding/decoding algorithms, we raised the question on the relevance of such schemes for compression purposes. Though, up to our knowledge, little has been made on the subject, we review some of the studies interested in this issue.

2.3 Compressing the neural data

Coding the neural data is an issue of growing interest, especially in the Brain Machine Interface (BMI) community. Indeed, several BMI applications require communication protocols between the neural living tissue and a rehabilitative prosthetic device. Electronic chips are devoted to collect neural spiking signals output and transmit it, mainly through a wireless channel. In that case, for power consumption reasons there is a need to compress data.

In [PPS05, CPK+07, RPP07], authors aim at compressing the waveform originated by each neuron separately. The output response is considered as a continuous function of time $s(t)$, an example of which is shown in Figure 7. As living neurons are connected to an acquisition device, this signal is obviously raw digitized according to the sampling frequency of the device. The raw of samples is chopped into adjacent, non overlapping, k-samples vector. This can be seen as a non-overlapping sliding window analysis. Having this representation of the signal as a series of k-dimensional vectors, a vector quantization (VQ) method is applied to compress it. Readers interested in (VQ) may refer to [GG92].

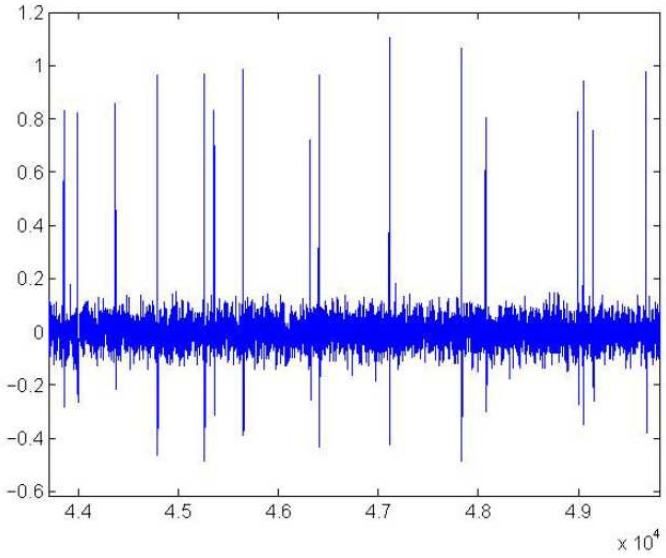


Figure 7: An example of spike train signal instance as output from a single neuron (from [RPP07]).

The (VQ) algorithms allow authors to map each k-samples vector, x_i, into an integer value c_i also called codeword among L possible values. The set of these L-codewords forms the so-called codebook C of the coder. Each codeword $c_i \in C$ is an index that stands for a k-dimensional vector, y_i, representative of a given class of possible x_i's. The codebook is constructed by a learning process as to have more y_i's among dense x_i population in $\mathbb{R}^k$. The only data transmitted is the codeword c_i compressed with a lossless coder. The signal can then be recovered through a lookup table (LUT) stored in the recipient chip, mapping each c_i into the equivalent y_i vector. Three different algorithms for the construction of the set of y_i's, also called the dictionary, are presented.

In [PPS05] the self-organizing map (SOM) algorithm is used for the construction of the coding dictionary. SOM is a neural network based algorithm. The base principle is a competition between neurons of the network. For any x_i presented to the input of the network, only a winning neuron fires, giving the nearest representative y_i in the sense of L^1 norm. The rest of the coder operates as described above for VQ coders. Interested readers may refer to [Koh98] for

further details about the SOM algorithm. According to authors SOM ensures a compression ratio of 32 : 1 for a codebook of 64 codewords with the quality measure, signal to noise ratio (SNR), equal to 14.7 dB in the regions of $\mathbb{R}^k$ mostly populated with spikes.

In [CPK+07], an improved version of the SOM algorithm is presented including a dynamic learning procedure. The goal of it, is to better map sparse regions of the input space. This is of crucial importance as spiking data is sparse and thus loss of information in sparse regions may originate large errors. Indeed, further steps of the application processing, as spike detection and sorting, may be altered if vectors in sparse regions are not well rendered. This method allow 42 : 1 compression rate for a codebook of 64 codewords with an SNR of 16.5 dB.

In [RPP07], an other classical VQ algorithm is tested, namely the weighted LBG quantizer. The algorithm is a recursive implementation of weighted k-means algorithm. Its basic principle in an incremental construction of the codebook by learning on a data of actual neural spiking signals. Indeed codewords are split in two at each stage and the codebook is updated in accordance at the next stage, this until a maximal distortion is reached in the sense of a weighted L^2 norm (see [RPP07] for an outline of the algorithm).

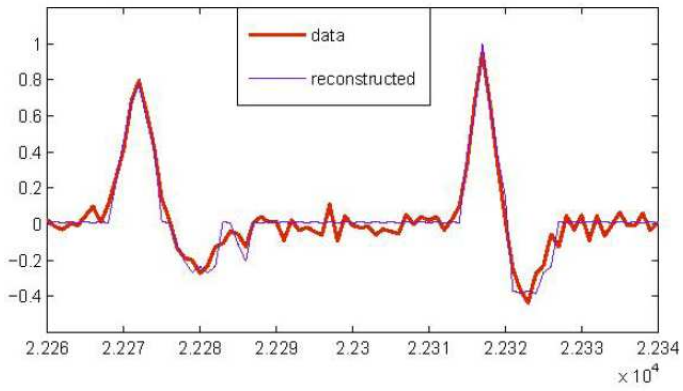


Figure 8: An example of a sample spiking signal reconstructed after weighted LBG quantization (from [RPP07])

In [NTACB07], authors propose an alternative to direct VQ compressing of the signal. Like in former methods, the input is the analog signal recorded as the output of a single neuron. The difference is that authors do not directly quantize the signal, they first make a wavelet analysis. The wavelet basis is constructed by scaling and translating a special finite-energy function, the mother wavelet. The authors propose the Daubechies-3 function as the mother wavelet for its similarity in shape with a spike waveform. The data originated by this analysis is a set of translation and scaling factors (t_i, s_i), defining a vector in the basis, and the amplitude, c_i, of the corresponding projection coefficient. A thresholding is made to eliminate non significant coefficients. Because of the sparseness of a spiking signal, only a few non-zero amplitude coefficients then remain. Bi-

ologically speaking, the translation t_i factor determines the time of occurrence of a spike, and the (s_i, c_i) couple is the shape signature of a spike. As there is a small amount of typical spikes signatures, authors constructed a reduced dictionary of the most probable signatures. Each (s_i, c_i) is then mapped into the nearest signature available in the dictionary. The only data that is transmitted is then the set of couples (t_i, d_i), where d_i is the index of the signature in the dictionary. An example of signal reconstruction after a compress/uncompress process is shown in Figure 9. This algorithm allows a compression ratio gearing from $1:30$ upto $1:80$.

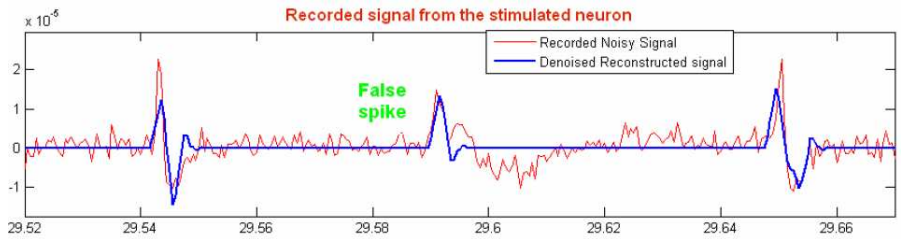


Figure 9: An example of a sample spiking signal reconstructed after vocabulary-based wavelet analysis (from [NTACB07])

The algorithms presented have considerable limitations. The major one is that neural data is considered as one-dimensional signal with no biological specificity. Besides each cell output is considered marginally as no neighborhood interactions are taken into account.

The following sections describe a coding scheme aimed at generating a spike-like code for an image stimulus, then representing it as a sparse data set, and finally compressing it. The decoding is also described. Up to our knowledge, this work represents the first attempt to assemble the three aspects of neural coding components (generating it, cracking it, and compressing it), for static image compression applications.

3 Rank order coding in the retina

Here we focus on Thorpe et al hypothesis which states that considering only the first wave of spikes could give a good idea of the message contained in spike trains [Tho90, TFM96, RT01, RT02]. We chose this model for its ability to code/decode a static image stimulus, with visually satisfying faithfulness as well as for its relative low computational cost.

Neurophysiologic evidence sustains this model. In [Tho90, TFM96, RT01, RT02], the authors showed that still image categorization can be realized by the visual cortex within very short latencies of about 150 ms or even faster. Such short responses can only be explained by a specific architecture or specific information encoding schemes. Thus the idea of rank order coding (ROC) was proposed. ROC is based on the assumption that: the neural information

is encoded by the order in which each ganglion cell emits its *first spike* from stimulus onset. Indeed, most excited neurons not only emit more spikes but also fire sooner. So, the rank order coding can be an explanation for visual system performance in ultra-fast categorization. Starting from the ROC assumption, a model of a retina generating spikes was proposed By Van Rullen and Thorpe [RT01]. This leads us to use the Thorpe retinal model as a first attempt to conceive a bio-inspired still images codec.

In the following, we give an overview of the neurophysiologic experimental results at the origin of this work. These experiments explore the coding mechanisms in the early stages of the primate visual system, especially in the retina. One of the major results shown is that biological image coding allows for recognition tasks to be undertaken within short latencies, around 150 ms [TFM96]. We then present a retina model aimed at explaining this human fast recognition capability [Tho90, TFM96, RT01, RT02]

3.1 Coding the first wave of spikes

Since the 60's, neuroscience has been interested in the coding mechanisms underlying the processing of visual stimuli in the primate's brain. The widely admitted assumption is that neurons firing rate conveys the information about the stimulus. Recently alternatives to rate coding emerged. In this context, Oram and Perrett studied face recognition capabilities of primates in their 1992 work [OP92]. The evidence was made that the decision occurs quickly though information has to be processed through several stages in the brain. As a consequence authors raised the idea of *the first wave of spikes* coding for the stimulus. This means that visual cortical areas consider only the first spike emitted by each neuron for face recognition tasks.

This result was generalized by Thorpe et al in [TFM96]. Authors confronted the ultra-rapid categorization hypothesis to stimuli that are not specific to face recognition. Stimuli considered for the experiment are natural complex scenes. The primate subject is given a go/no-go task, namely it has to answer the question whether there is animal in the scene or not. Pictures are flashed during 20 ms and output signals (Event Related Potentials) are recorded in the frontal sites of the brain. The study of the output data shows that decision occurs within around 150 ms. Later this result was extended to non biologically relevant targets in [VT01]. As an explanation authors claimed that the first spikes wave is the code used for ultra-rapid categorization tasks.

Recently Meister et al made an attempt validating the relevance of first spikes wave code by quantitative measures [GM08]. Authors investigate the neural output of the salamander retina. Stimuli considered are flashed still images of gratings. Spike trains are recorded at the level of the retina ganglion cells. Two metrics are then measured over spike trains. The first one is the first spike latencies, the second is the spike firing rates given a time window of fixed size. The experimental results confirmed that first spike latency time is more discriminative for the input signal. Besides an amount of information metric, the entropy (see [CT91] for entropy definition), was computed and shows that first spike latency conveys twice as much information as firing rate. This val-

idates plausibility of first spike latencies predominance in the visual neural code.

As described in this section, neurophysiologic experiments support the idea that: *the first wave of spikes convey most of the information in the neural code.* Based on this assumption, Thorpe introduced in [Tho90, TFM96, RT01, RT02] a coding process, the ROC coding. In the following, we describe the ROC-based algorithm, specified in [RT01], to model this approach.

3.2 Rank order coding principle

Thorpe et al introduced the ROC coding approach to explain primate's visual system performances for tasks involving massive processing as categorization. Experiments show that we are very talented in categorizing images even with very short presentation durations. As an explanation for this extraordinary performance in ultra-fast categorization, Thorpe et al [Tho90, RT01] proposed that the order in which each ganglion cell emits its first spike codes for the stimulus. This originated the ROC code. ROC relies on the following simplifying assumptions:

i) From stimulus onset, only the first spike emitted is considered in the response.

ii) The time to fire of each ganglion cell is proportional to its degree of excitation.

iii) Only the order of firing of the neurons encodes for the stimulus.

Thorpe states that transmitting the order and only the order by which neurons fire their first spike is sufficient to extract relevant information about the stimulus. The lacking information is recovered through a priori known data. This code is the basis of a conceptual coding retina implementation. This retina aims at reproducing a biology-like code by generating a series of spikes. The specification of ganglion cells behavior as well as the retina architecture is described in the following section.

3.3 Thorpe retina model proposal

In this section we show how a still image is decomposed then reconstructed with Thorpe bio-inspired retina model [RT01]. We then raise the analogy between this retina functioning and signal processing algorithms.

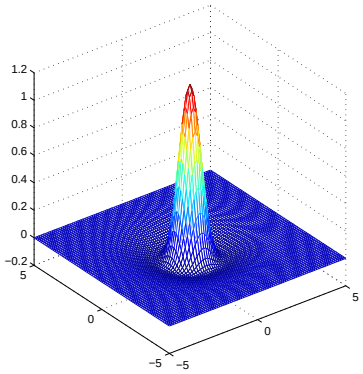
3.3.1 The Thorpe retina bricks

Here we review the main elements and principles involved in the virtual retina functioning. We first investigate the behavior of the ganglion cells. These cells tile the entire surface of the retina and have receptive fields of different sizes. Several studies stated that ganglion cells are analogous to linear filters [Rod65, Fie94]. In [Fie94] the *dog* filters are cited as a good approximation to reproduce ganglion's behavior. The *dog* filter is a weighted difference of Gaussians and is defined as follows:

$$dog(x,y) = w_c g_\sigma(x,y) - w_s g_{\alpha\sigma}(x,y), \tag{19}$$

where σ is the so-called central standard deviation of the filter, α is an a priori fixed real number, g_σ (resp. $g_{\alpha\sigma}$) is the Gaussian kernel of standard deviation σ (resp. $\alpha\sigma$), and w_c (resp. w_s) is the weight of g_σ (resp. $g_{\alpha\sigma}$). With biologically realistic parameters a *dog* filter applied to the image is a contour detector as shown in Figure 10.

(a)



(b)

(c)

Figure 10: Behavior of the *dog* filter when applied to Lena with biologically plausible parameters. (a): the classical test image Lena. (b): the *dog* filter with parameters $\alpha = 3$ and $w_c = w_s = 1$ and $\sigma = 0.5$. (c): the absolute value of the image resulting from the convolution of Lena with the filter in (b).

For given weights w_c and w_s, the value of σ determines the spatial resolution of a *dog* filter. In [RT01] authors considered 8 possible σ's, i.e., 8 possible scales, for the filters. This specific choice of scales number is arbitrary and has an incidence on the quality of the image reconstruction as we will show later in this paper. These scale-varying filters chop the image spectrum into different subbands. As shown in Figure 11, *dog* filters thus chosen cover a large part of the available spectrum.

For each scale the receptive fields of ganglion cells, here modeled by the *dog* filters, have to tile the input image space. Thus filters are distributed according

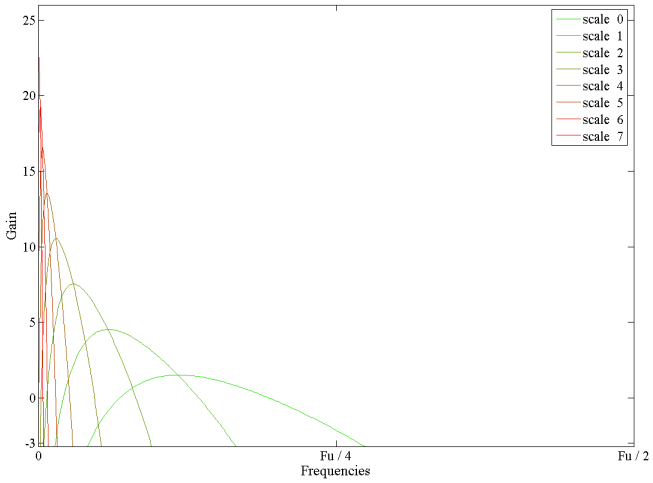


Figure 11: Spectra of the *dog* filters chosen in the Thorpe retinal model. The horizontal axis represents the frequencies from 0 to $\frac{F_u}{2}$ where F_u is the image sampling frequency. The vertical axis represents the gain of the filters in dB.

to a regular layered grid G. All filters of the same layer G_l of G have the same scale s_l, and G_l density is inversely proportional to s_l. G is said to be a finite dyadic grid. Such a grid is represented in Figure 12. Formally a finite dyadic

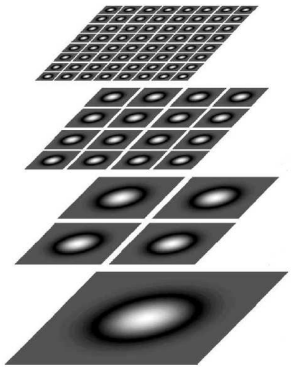


Figure 12: Dyadic repartition of the *dog* filters considered for image analysis. Here 4 scales are represented. As we get from the tightest scale (at the top layer) to the largest one (at the bottom layer) the number of filters considered per scale is divided by 4 each step (from [RT02]).

grid is defined as in [CCG+06]:

Definition 1 ***Finite dyadic grids*** *: A finite dyadic grid G has an obvious unique representation as a finite 0-4 tree, a binary tree where every node (which represents a cell of G) has either 0 or 4 children nodes. Conversely, every 0-4 tree defines a finite dyadic grid.*

In a layer G_l cells are regularly spaced by 2^{s_l} pixels over the horizontal and vertical axes. For convenience scales are numbered from 0 (for the tightest scale) to 7 (for the largest one). The set of cell locations defines a subset U of $\mathbb{N}^3$. For an N^2-sized image, U is defined by:

$$U = \{(s,i,j), \setminus (s,i,j) \in \mathbb{N}^3, s \in [0,7], i < N, j < N, i \equiv [2^s] \,\text{and}\, j \equiv [2^s]\} \tag{20}$$

Along with U, we define an undersampling function u over $\mathbb{N}^3$ by:

$$u(s,i,j) = \begin{cases} 1 & \text{if} \quad (s,i,j) \in U \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Each cell C_k in the grid, thus delineated, is defined by its scale s_k, and its position vector (i_k, j_k) in the image space. Then, the $dog_{s_k i_k j_k}$ filter of the cell C_k is expressed as follows:

$$dog_{s_k i_k j_k}(x,y) = w_c g_{\sigma_{s_k}}(x - i_k, y - j_k) - w_s g_{\alpha\sigma_{s_k}}(x - i_k, y - j_k) \tag{22}$$

where $\sigma_{s_{k+1}} = \frac{1}{2}\sigma_{s_k}$.

Remark It is to be mentioned that such an architecture is not biologically realistic for mainly two reasons. First, in [Ham74] experiments made on cat's retina showed that the largest receptive fields of ganglion cells are, in average, only 2 to 3 times wider than the tightest ones. Whereas in the model presented the scaling factor goes up to 128 from layer 0 to 7. Second, for a given scale, ganglion cells are not regularly dispatched over the retina. In fact observations made on primates [LKY98] and cats [Ham74] showed large receptive fields to be more dense in the periphery of the retina while tight receptive field cells are more dense in the center.

Having the family of filters $(dog_{s_k i_k j_k})$ described above and a static image to code we show in the next section how to model the generation of spikes as the retina output.

3.3.2 Image decomposition

We consider the layered dyadic grid G of filters described in the previous section and an input image f to analyze. In order to measure the degree of activation of a given neuron, we compute the convolution of the original image f by a *dog* filter defined by its scale s and its location (i,j), so that:

$$c_{sij} = \sum_{x,y=-\infty}^{\infty} dog_s(i-x, j-y)\, f(x,y). \tag{23}$$

Neurons responses are then sorted in the decreasing order of their amplitude, i.e., $|c_{sij}|$. For an N^2-sized image we obtain the series of N_s sorted triplets $(s_k, i_k, j_k)_{0 \leqslant k \leqslant N_s - 1}$ defined recursively by:

$$\begin{cases} U_0 & = U \\ (s_0, i_0, j_0) & = \underset{(s,i,j) \in U_0}{\operatorname{argmax}} (|c_{sij}|) \\ U_k & = U \setminus \{(s_0, i_0, j_0), ..., (s_{k-1}, i_{k-1}, j_{k-1}))\} \\ (s_k, i_k, j_k) & = \underset{(s,i,j) \in U_k}{\operatorname{argmax}} (|c_{sij}|) \end{cases}$$

A step further, the authors in [Tho90, TFM96, RT01, RT02] showed that there exists a mapping between the rank k, and the amplitude $|c_{s_k i_k j_k}|$ which has approximately the same shape across natural images. Thus authors supposed that the exact values of $|c_{s_k i_k j_k}|$ are known a priori at the level of the decoder. This allows a great gain in the amount of information necessary to encode the input image. The characteristic series of $|c_{s_k i_k j_k}|$ is constructed off-line and stored in a look up table (LUT). The LUT used in [RT01] is obtained as an average over a set of natural images. Figure 13 shows some examples of LUT's that could be used for coefficient recovery.

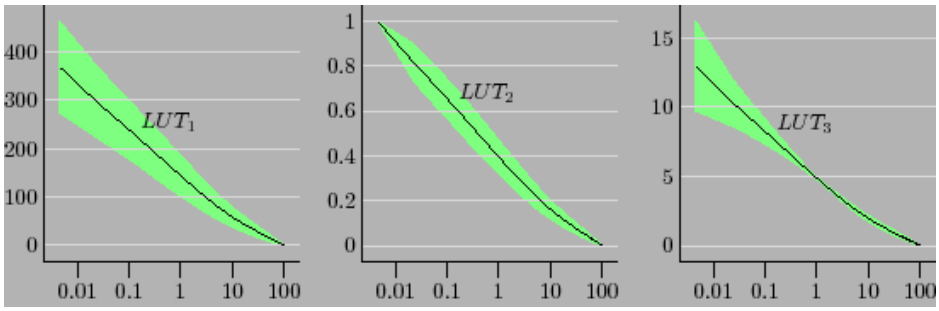


Figure 13: 3 examples of possible LUT's for the activation coefficients recovery. The vertical axis represents estimated values of the coefficients. The horizontal axis represents the percentage of fired spikes. The green area around the curve represents the standard deviation over the set of tested images (from [Per03]).

As the LUT infers only the value of $|c_{s_k i_k j_k}|$, we obviously need to store the series $(sign(c_{s_k i_k j_k}))$ to recover the activation coefficients.

At the end of this stage, this virtual retina generates a sorted series (s_k, i_k, j_k) of neurons and the corresponding activation signs $(sign(c_{s_k i_k j_k}))$. We present in the next section the way adopted in [RT01] to reconstruct the original stimulus with no other data than the series of N_s quadruplets (e_k) defined by:

$$\forall\, 0 \leqslant k \leqslant N_s - 1,\ e_k = (s_k, i_k, j_k, sign(c_{s_k i_k j_k})). \tag{24}$$

3.3.3 Image reconstruction

This section treats the way rank order code is decoded to recover the original image signal under the Thorpe model assumptions.
First let us suppose we know the c_{sij} coefficients. In addition, we make the simplifying assumption that all filters in the dyadic grid are orthonormal. In this case the reconstruction estimate $\tilde{f}$ of the original input f is obtained, for N_s emitted spikes, as follows:

$$\tilde{f}(x,y) = \sum_{k=0}^{N_s-1} c_{s_k\, i_k\, j_k}\, dog_{s_k}(i_k - x, j_k - y), \tag{25}$$

Remark Considering that, the *dog* filters form an orthonormal basis, is an approximation as mentioned in [SF07]. Instead, we have:

$$< dog_{s_k,i_k,j_k}, dog_{s_l,i_l,j_l} >= \varepsilon << 1,\ \forall\, (s_k, i_k, j_k) \neq (s_l, i_l, j_l). \tag{26}$$

Some enhanced versions of this algorithm are presented in the literature. For example, in [PST04], the authors use a matching pursuit procedure in order to eliminate redundancies between coefficients; and in [SF07], the authors proposed to invert the *dog* basis using a Moore-Penrose pseudo-inversion procedure for the same purpose.

To recover the missing coefficients, we use a (LUT) as specified earlier. In this paper, instead of a predefined mapping as in [RT01], a parametric function LUT_γ of the rank k of e_k is defined, as in [Per03], by:

$$LUT_\gamma(k) = C\, k^{-\gamma}, \tag{27}$$

where C is an arbitrary constant positive value and γ is the parameter of the function. The motivation is to find the best fit for each image. To do so, we implemented a gradient descent over γ that minimizes the criterion of Mean Squarred Error (MSE) between the real coefficients $|c_{s_k i_k j_k}|$ and the estimated ones $LUT_\gamma(k)$. The optimal LUT parameter will be denoted by γ_0. Figure 3.3.3 shows a comparison between actual coefficients and the LUT_{γ_0} functional for Lena.

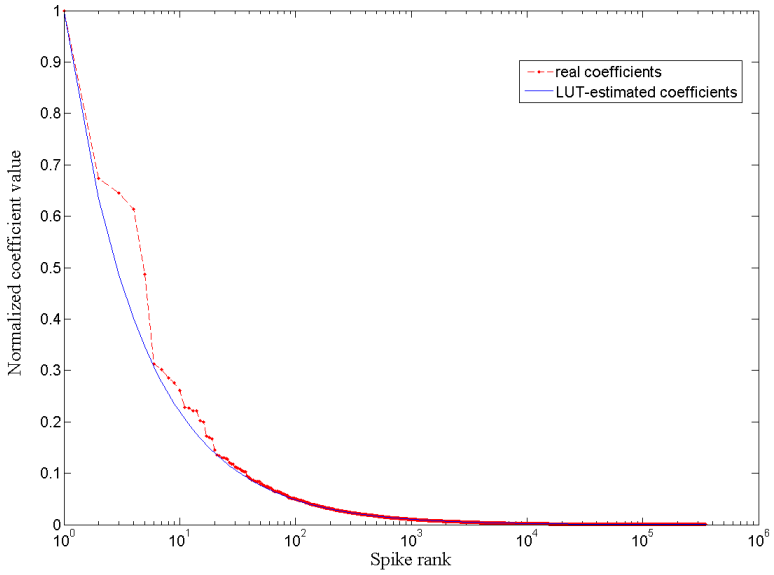


Figure 14: LUT for Lena: The dotted line of real activation coefficients values obtained compared to the functional LUT_γ (continuous line)

Reconsidering the estimation of $\tilde{f}$, replacing (27) in (25) the reconstruction formula (25) becomes:

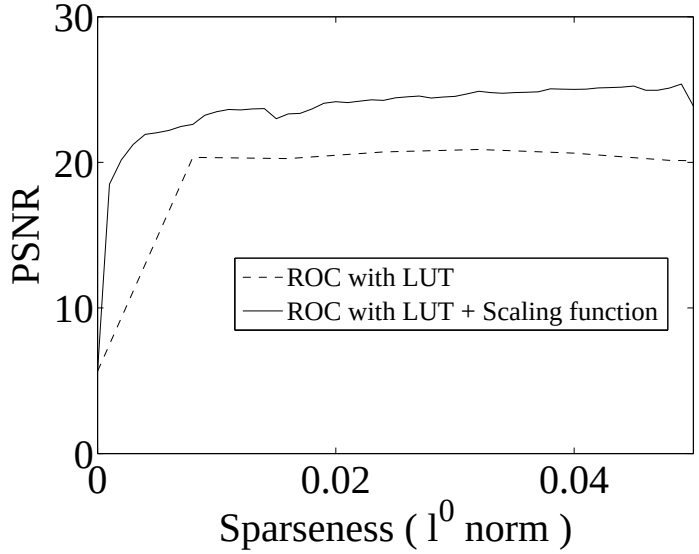
$$\tilde{f}(x,y) = \sum_{k=0}^{N_c-1} sign(c_{s_k i_k j_k})\, LUT_{\gamma_0}(k)\, dog_{s_k}(i_k - x, j_k - y).$$

This formula allows progressive reconstruction as one can stop at a given maximum rank value, N_s. An example of progressive reconstruction is shown in Figure 15. This property allows the coder to be scalable.

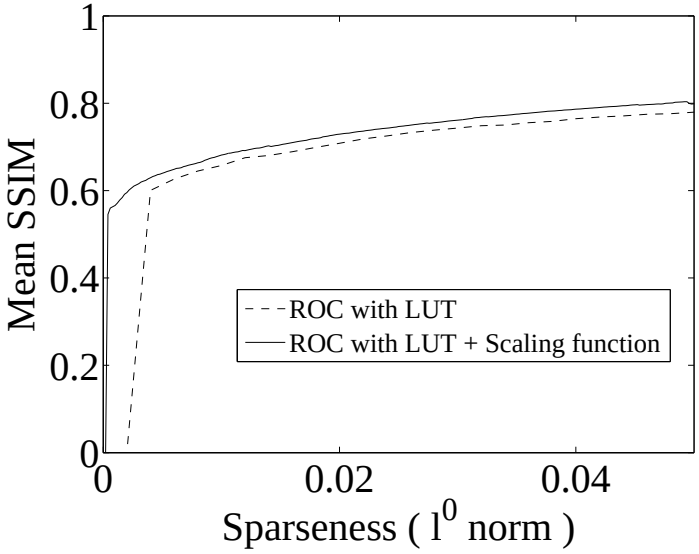
Remark As an enhancement to the current coder, we added a Gaussian scaling function [Mal89] to catch low frequencies omitted by the original model. The latter modification improves the image reconstruction quality by $5\,dB$ in PSNR for 5% of fired spikes used for the reconstruction (see Figure 15 for a qualitative comparison). Figure 16 shows the gain in quality provided by this modification in terms of PSNR and mean SSIM [WBSS04].

It is to be noted that the reconstruction evolution maps actual retina behavior, as low frequencies are first transmitted, then details progressively added.

Figure 15: Progressive image reconstruction of Lena using Thorpe ROC coder with a scaling function. As the percentage of spikes decoded increases, the visual quality of the coded/decoded images increases. Lower right: original ROC reconstruction with no scaling function.



(a)



(b)

Figure 16: Comparison of reconstruction of Lena (cf. 10(a)) quality in the case of the use (solid line)/no use (dotted line) of a Gaussian scaling function. The horizontal axis represents the percentage of spikes considered for image reconstruction. The vertical axis represents image quality (PSNR in 16(a) and SSIM in 16(b)). The addition of a scaling function to the set of *dog* filters do increase reconstruction quality

4 Coding the spikes as generated by the virtual retina

We introduce in this section an end-to-end lossy coder/decoder for still images based on the retina model of section 3. The virtual retina we considered generates a sorted series of spikes, modeled by the series (e_k) (cf. (24)). The issue addressed is how to represent this data in order to code it for the best, i.e., transmitting as much information as possible under a bandwidth constraint. As to answer this question we first, outline the novel codding scheme we proposed, then we define a sparse representation for the spiking code, and finally we present the arithmetic coding algorithm used to compress the spikes thus represented.

4.1 System overview

Our compression scheme encompass three stages: first, a spiking retina model based on rank order coding (ROC), second, a zero-run length coder, namely a stack run coder, and finally an arithmetic coder. This is summarized in the block diagram in Figure 17. The following sections detail each one of these stages.

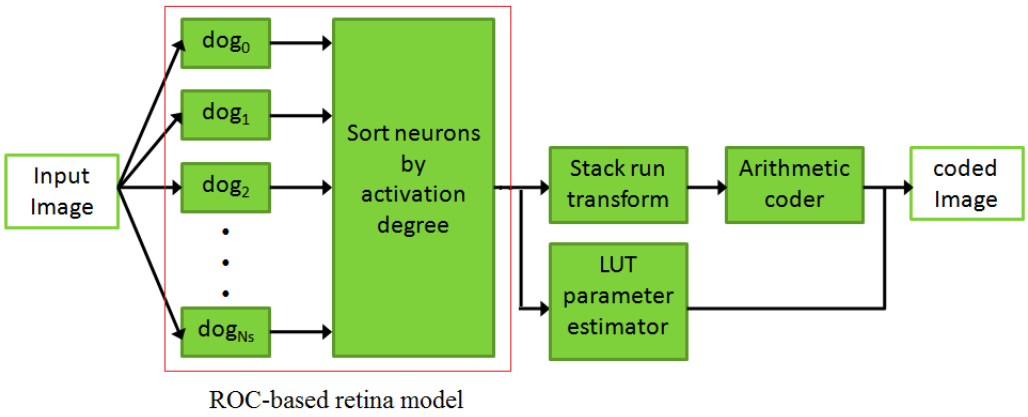


Figure 17: Block Diagram of the compression scheme. First, a ROC based retina model. Second, the ROC code is zero-run length encoded by the stack run coder. Finally a first order arithmetic coder is applied to get the compressed image file. The decoding process goes exactly the opposite way.

4.2 The Bitstream code for a static image using ROC

Let us reconsider the series of spikes, (e_k) in (24), and the optimal parameter γ_0 in (28). Applying (28), we can obtain a reconstruction $\tilde{f}$ of the coded image f up to a translation and scaling factor. Indeed the analyzing *dog* filters of the retina are invariant to the mean value m of the image f. Reconsidering

equation (23) for the calculus of neurons activation coefficients c_{sij} we have :

$$\begin{aligned} c_{sij} &= \sum_{x,y=-\infty}^{\infty} dog_s(i-x, j-y)\, f(x,y) \\ &= \sum_{x,y=-\infty}^{\infty} dog_s(i-x, j-y)\ (f(x,y)+m)\ \forall m \in \mathbb{R} \end{aligned} \tag{28}$$

Thus the code transmitted to the decoder is also invariant to m. Besides if we consider the whole coding/decoding system as a black box h such that $\tilde{f} = h * f$, then h has a non-unitary gain G_H. Hence in order to a good estimate of f, $\tilde{f}$ image must be rescaled and translated. For this, once can transmit m and G_H values or $min(f)$ and $max(f)$, the boundary values of f.
The bitstream we get is then as follows :

1) $min(f)$

2) $max(f)$

3) γ_0

4) the series of Spikes

The next section details the way we represent the spike series in the bitstream coding for the image f.

4.3 Spikes coding using Stack Run algorithm

The ROC coder presented in Section 3.3 generates a series of at most N_s spikes denoted as $(e_k)_{0 \leqslant k \leqslant N_s - 1}$, where N_s is the size of the dyadic grid. As demonstrated in Figure 15, few spikes can reasonably represent the image to code. This property is loosely referred to as sparseness in the literature. The sparseness of neural codes has been shown to help conceive meaningful representations of data [OF96]. Keeping sparseness as a design principle, we define, in this section, a sparse representation of the series (e_k) in the sense that, we look for a series mainly populated with zero's. We then introduce a zero-run length coder well suited for sparse data coding, namely the stack-run coder.

We consider the definition of e_k in (24). For each triplet elements (s_k, i_k, j_k) of e_k, let us define the scalar index $r_k \in [0, N_s - 1]$ by:

$$r_k = s_k N^2 + \frac{i_k}{2^s} N + \frac{j_k}{2^s}, \tag{29}$$

such that, $c_{s_k i_k j_k}$ can be denoted c_{r_k}. So, the ROC response restricted to the first N_c spikes can be written as the following sorted list of N_c couples (spiking cell index, sign of the response):

$$M_{retina}^{N_c} = ((r_0, sign(c_{r_0})), ..., (r_k, sign(c_{r_k})), ..., (r_{N_c}, sign(c_{r_{N_c}}))).$$

Coding this list may be quite expensive in terms of bit-rate. This is because from one index to the next, one does not consider their spatial arrangements. The dimension of M_{retina} is N_c but the values to encode may be large. For

this reason, we propose instead a representation taking into account the relative positions of the spiking cells. So we define the vector:

$$M_{sparse}^{N_c} = (m_0, ..., m_l, ..., m_{N_s-1}),$$

so that:

$$m_l = \begin{cases} k\, sign(c_{r_k}) & \text{if } \exists\, k < N_c \ / \ l = r_k \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

$M_{sparse}^{N_c}$ is equivalent to $M_{retina}^{N_c}$ up to a one-to-one map. In fact we verify that:

$$\forall\, 0 \leqslant k \leqslant N_s - 1,\ \exists l_0 \text{ such that } e'_k = (l_0, sign(m_{l_0})),$$

where:

$$\forall\, 0 \leqslant k \leqslant N_s - 1,\ l_0 = \underset{l \in [0..N_s-1]}{\arg} (|m_l| = k). \tag{31}$$

The dimension of $M_{sparse}^{N_c}$ is N_s (the size of the dyadic grid), but the range of values to encode now depends on N_c. For a sufficiently small value of N_c, $M_{sparse}^{N_c}$ is a sparse data set: this is the feature we tried to enhance in our message code $M_{sparse}^{N_c}$.

Zero-run length coders are well suited for the compression of such data sets [Say00, BWC90]. In our scheme we use an enhanced run-length coding algorithm, the stack run code [TVC96]. Stack run coding uses a 4-ary dictionary $\{0, 1, +, -\}$. $M_{sparse}^{N_c}$ is mapped into a series of couples: (zero-run length, non-zero value). As specified in [TVC96] the subsequent bit-wise operations are applied:

i) "+" represents binary bit 1 in run lengths.

ii) "-" represents binary bit 0 in run lengths.

iii) "0" encodes a binary bit value of 0 in non zero values.

iv) "1" encodes binary bit 1 in non zero values.

The use of 4 symbols in the alphabet removes ambiguity between run lengths and coefficient values. We then modify the code using the following additional rules :

i) Every non zero value is set to its absolute value after we stored its sign.

ii) The MSB of a non zero value is set to "+" if the sign is positive.

iii) The MSB of a non zero value is set to "-" otherwise.

For example, if we consider the code:

$$M_{sparse}^{N_c} = (0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, -18), \tag{32}$$

then applying the rules of stack-run encoding we obtain the code M with:

$$M = +\,+\,+10+\,+\,-\,+0100-, \tag{33}$$

where the first $+\,+\,+$ string encodes for 7 zero's and $10+$ for 5, etc. It is to be noted that the stack-run thus defined is a simplified version of the implemented algorithm. Further optimizing mechanisms are discussed in [TVC96].

Having the message S_{stack} to code, we can apply an arithmetic coder for compression with optimal performances. Basic features of arithmetic are presented in the following section.

4.4 Arithmetic coding

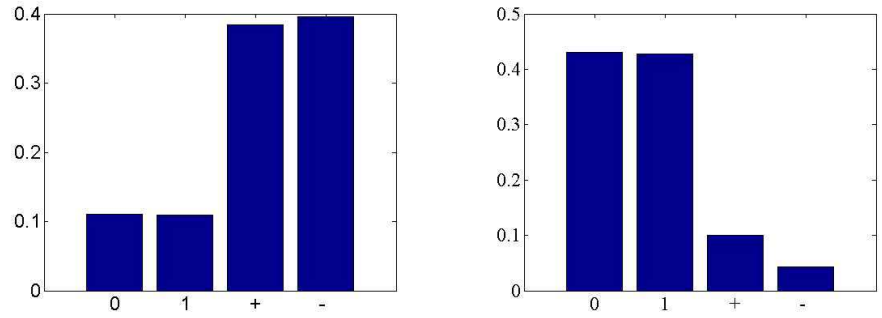
While reading the code in (33), the decoder can switch from the *run length context* to the *non-zero value context* as it encounters the character 0 or 1. Obviously the decoder switch contexts in the opposite way as it encounters the character + or −. This remark allows us splitting the code we obtained into 2 different files, one for the run length context and the other for non-zero value context. Considering the example in (33), we get in the first file F_0:

$$F_0 = +++1+-+0, \tag{34}$$

and in the second one F_1:

$$F_1 = 0+100-. \tag{35}$$

Thanks to this split, we get a first file F_0 mainly populated with +'s and −'s and F_1 mainly populated with 0's and 1's. Having these two files with enhanced contextual features, we can apply an arithmetic coder, [Say00], which we said to be a first order coder, for compression with optimal performances. Indeed, when we separate zero run lengths context from non zero values context, we skew occurrence probabilities of the symbols in each file. Figure 18 shows the occurrence probability of each symbol from the alphabet A in the files encoding Lena. This is useful as arithmetic coding is an efficient lossless compression algorithm when dealing with small alphabets and highly skewed probabilities [Gir93, Say00].



(a) Histogram of the zero run length file

(b) Histogram of the non zero values file

Figure 18: Histograms of the two files composing the stack run code

Now let us detail arithmetic coding algorithm. Arithmetic coding is a data compression technique that encodes data by creating a code string which represents a fractional value on the number line between 0 and 1 [Say00, Jr.84]. The message $S = (s_0, s_1, .., s_i, ..., s_n)$ we manage to encode is a series of symbols from the 4-symbol alphabet $A = \{0, 1, +, -\}$. Unlike entropic coders, arithmetic ones do not replace each symbol s_i in S by a predefined sequence of full bits. It encodes the whole message at once. For this to be done we are given a model distribution M, ie a one-to-one map that associates each symbol of the alphabet A to its occurrence probability in the message S. Formally a model M is a function defined as :

$$M : A \longrightarrow [0, 1[: \ a_i \mapsto P_M(a_i),$$

where $P_M(a_i)$ is the occurrence probability of the symbol $a_i \in A$ in a given message [BCK07]. Furthermore we define a function F_{XM} along with P_M by :

$$F_{XM}(a_i) = \sum_{k=0}^{i} P_M(a_i), F_{XM}(a_{-1}) = 0. \tag{36}$$

The basic guidelines of the algorithm lie on the association of a string of symbols to an unique interval in $[0, 1[$. First we split the interval $[0, 1[$ into 4 subintervals $(T_i^0)_{0 \leqslant i \leqslant 3}$ such as their lengths are proportional to $(P_M(a_i))_{0 \leqslant i \leqslant 3}$. These subintervals are of the form $T_i^0 = [F_{XM}(a_{i-1}), F_{XM}(a_i)[$. If $s_0 = a_i$, we associate T_i^0 to the substring message (s_0). In the second step the subinterval T_i^0 is split to its turn into subintervals $(T_i^1)_{0 \leqslant i \leqslant 3}$ which lengths are proportional to $(P_M(a_i))_{0 \leqslant i \leqslant 3}$. In the same manner we associate a subinterval T_i^1 to the string message (s_0, s_1) corresponding to the value a_i of s_1. The same process goes on recursively until we reach the end of the string mes sage $S = (s_0, s_1, ..., s_n)$. At the the end of the process we get an interval T_i^n coding for the whole message at once. The median value of T_i^n could be used as a tag encoding for the message. Figure 19 shows an example of the processing of the arithmetic coding algorithm.

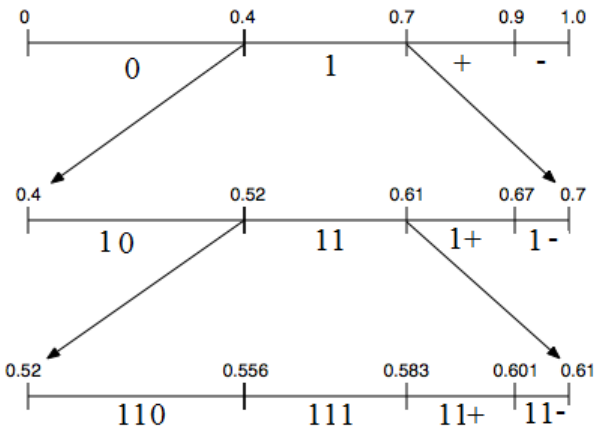


Figure 19: An example of arithmetic coding : The string to encode is 11+ and the model M used is defined by $P_M(0) = 0.4$, $P_M(1) = 0.3$, $P_M(+) = 0.2$ and $P_M(-) = 0.1$. At the first step the interval $[0, 1[$ is split into subintervals which lengths are proportional to the model probabilities. We then associate the first symbol 1 to the interval $[0.4, 0.7]$. At the second stage we split $[0.4, 0.7]$ into intervals according to the model and the string 11 is associated to the interval $[0.52, 0.61]$. The interval $[0.52, 0.61]$ is split to its turn according to the model and the string 11+ is finally associated with the interval $[0.583, 0.601]$.

Having this coder thus specified we present in the next section a comparison of this coder to other image compression well established standards.

5 Results

At this stage we have a complete implementation of a coder/decoder for still images. To quantify the cost in terms of bandwidth of our coder we first formalize notions of amount of information and overview some information theory elements. As to compare our work to existing standards we define a quality metric that accounts for human perception then, plot comparative rate-quality curves. We finally discuss competitiveness of our codec compared to $JPEG$, $JPEG2000$ [SCE01, CSE00].

5.1 Preliminaries

5.1.1 Sparseness metrics

We previously referred to sparseness as a characteristic of data structures mostly populated with zero-valued elements. Sparseness helps conceive meaningful representations of data [OF96]. Thus we need measure the sparseness degree provided by the data representation adopted and check coding costs according to this sparsity. The most commonly used measure for noiseless data is the l^0 norm. The l^0 norm of a $k-$sized code vector X, denoted $\| X \|_0$, is defined as follows [KC03, HR08] :

$$\begin{aligned} \| X \|_0 &= \frac{1}{k} \sum_{0}^{k-1} \| x_i \|^0 \\ &= \frac{1}{k} \sharp\{i, x_i \neq 0\} \end{aligned} \tag{37}$$

In our case the l^0 norm of the rank order code sequence is the percentage of fired spikes.

5.1.2 Cost metrics

We want to define a measure of *the amount of information* necessary to convey the the image code. To do so, we delineate the notion of entropy H, then we compare the theoretic estimate of H to actual image rate, and verify the accuracy of this estimate.

We consider the two files F_0 and F_1 generated by the stack run coder (cf. Section 4.3). Both files are populated with symbols from the alphabet $A = \{0, 1, +, -\}$. Information theory provide tools to estimate the minimum bandwidth compulsory to transmit a given source file F_i, namely the entropy. The entropy is a quantity measured for a random variable. The unit used for this quantity is the bit per symbol. In our case two random variables X_0 and X_1 are considered. X_0 (resp. X_1) maps each next character drawing in F_0 (resp. F_1) to a given symbol a_i from the alphabet A. The histograms in Figure 18 are an estimate of the probability distribution functions P_{M0} and P_{M1} of these random variables. Having these estimated probabilities we compute the entropy or the so called Shannon entropy by [CT91]:

$$H(X_j) = -\sum_{i=0}^{3} P_{Mj}(a_i) log_2(P_{Mj}(a_i)), \tag{38}$$

where $j = 0, 1$. As we need $H(X_j)$ bits to encode each symbol in F_j, we can compute the theoretic bandwidth we need to convey all the file by a simple multiplication by the number m_j of symbols in F_j. As each couple of files encodes for a whole image we sum H_{X_0} and H_{X_1} then we divide by the number of pixels N^2 to get an entropy rate R in bits per pixel. We summarize this calculus in the following equation :

$$R = \frac{1}{N^2}(m_0 H_{X_0} + m_1 H_{X_1}) \tag{39}$$

Along with this rate measure, we verify the size of the encoded image on the file system. Figure 20 compares these two measures, for Lena as a functional of the percentage of fired spikes. As we can see the real size of the coded image and Shannon entropy estimate are close to each other. The real cost of the coded image is even lower than the theoretic bound of the entropy which means that redundancy has been correctly removed (see [CT91, Say00]).

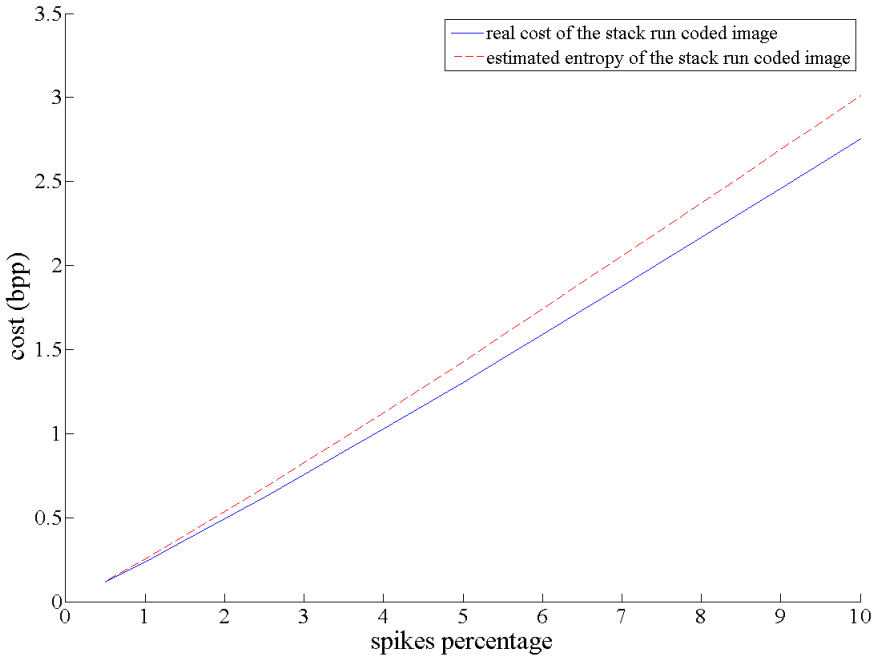


Figure 20: Cost of the stack run coded image 10(a). The straight line curve represents the cost of coding in terms of bits per pixel as a functional of the percentage of fired spikes. The dashed line represents the Shannon entropy of the same code in bits per pixel as a functional of the percentage of fired spikes.

5.1.3 Quality metrics

Now we need to quantify the quality of the reconstructed image $\tilde{f}$ compared to the original one f . We find in the literature several quantitative measures

for image reconstruction quality. The more common are the median square error (MSE) and the peak signal to noise ratio ($PSNR$). These quantities are computed through pixel to pixel differences and are defined as follows :

$$\begin{aligned} MSE(\tilde{f}, f) &= \frac{1}{N^2} \sum_{x,y=0}^{N-1} \| \tilde{f}(x,y) - f(x,y) \|^2 \\ , PSNR(\tilde{f}, f) &= 10\, log_{10} \left(\frac{{MAX_I}^2}{MSE(\tilde{f}, f)} \right), \end{aligned}$$

where N^2 is the image size and MAX_I is the maximum possible pixel value of the image f. In

PSNR and MSE are quadratic measures and thus are well suited for optimization processes. Besides MSE is easy to compute. These methods have proved to be inconsistent with human eye perception [Gir93]. Other tools quantify the reconstruction fidelity by comparing images structures. Among them the Structural SIMilarity (SSIM) index [WBSS04] is a measure for statistical similarity between two images $\tilde{f}$ and f. The SSIM index is a functional of the first and second order statistical metrics. The computation of it consists in computing a similarity index between two analogous windows in $\tilde{f}$ and f. The SSIM measure is then the average over all possible windows of the index. The resulting quantity is a coefficient between 0 and 1. Formally we define SSIM [WBSS04] by :

$$\begin{aligned} SSIM_{index}(W_x, W_y) &= \frac{(2\,\mu_{W_x}\mu_{W_y} + c_1)(2\,cov_{W_x\,W_y} + c_2)}{(\mu_{W_x}^2 + \mu_{W_y}^2 + c_1)(\sigma_{W_x}^2 + \sigma_{W_y}^2 + c_1)} \\ M_{SSIM}(\tilde{f}, f) &= \frac{1}{N_w} \sum_{W_x, W_y} SSIM(W_x, W_y) \end{aligned} \tag{40}$$

where W_x is a window of $\tilde{f}$, W_y the corresponding window in f, μ_{W_x} the average of W_x, μ_{W_y} the average of W_y, $\sigma_{W_x}^2$ the variance of W_x, $\sigma_{W_y}^2$ the variance of W_y, $cov_{W_x\,W_y}$ the covariance of W_x and W_y, N_w the number of windows.

5.2 Comparison to JPEG standards

We compared our results to the existing JPEG standards behavior under strong bandwidth restriction. Performances are comparable until 0.15 *bpp* image rate, which shows our algorithm to have encouraging performances. Comparison curves, in terms of PSNR and SSIM, are plotted in Figure 5.2.

Besides our codec shows good robustness to noise compared to JPEG and JPEG2000. Figure 22 shows the comparative performances of our codec and JPEG standards when dealing with noisy data. Indeed, the wavelet-like retina behavior in the model [RT01] enables a better robustness.

Rate/quality curves are plotted in Figure 23 and show up to 6 dB of gain in PSNR and 0.4 in mean structural similarity measure (SSIM) for 0.25 *bpp* of image rate compared to classic JPEG. As the rate increases JPEG codecs convey more high frequency (HF) signals, which are noise. This explains the decreasing rate/quality behavior of JPEG. As HF is encoded with loss in JPEG, artifacts appear in the image decoded. On the contrary, our new codec do not show artifacts because every spike is transmitted with no loss of information

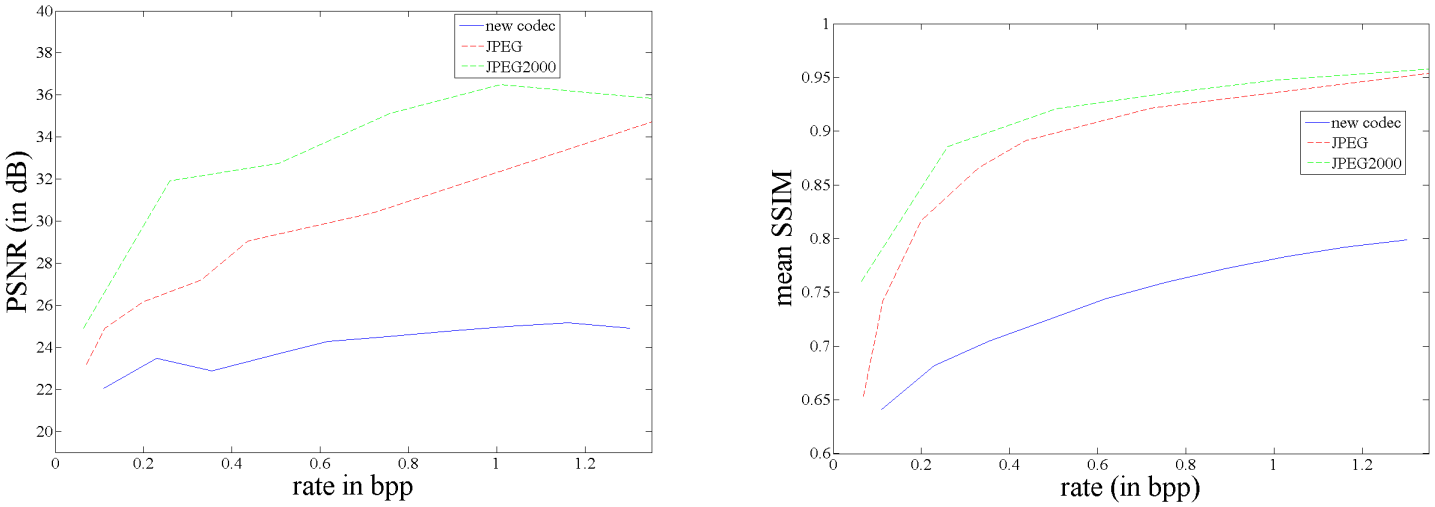


Figure 21: Comparison between JPEG, JPEG2000 and our new codec (PSNR on the left, mean SSIM on the right).

and encodes for the whole image. The scalability of our codec is monitored only by the choice of the number N_c of spikes to be encoded.

Figure 22: Robustness to noise: qualitative comparison between our new codec, JPEG, and JPEG2000 under the same rate restriction (here 0.27 *bpp*). Upper left: Lena (renormalized in the range of values from 0 to 1) with additive Gaussian noise ($mean = 0$, $variance = 0.05$). Upper right: coded/decoded image using the new codec. Lower left: coded/decoded image using JPEG. Lower right: coded/decoded image using JPEG2000.

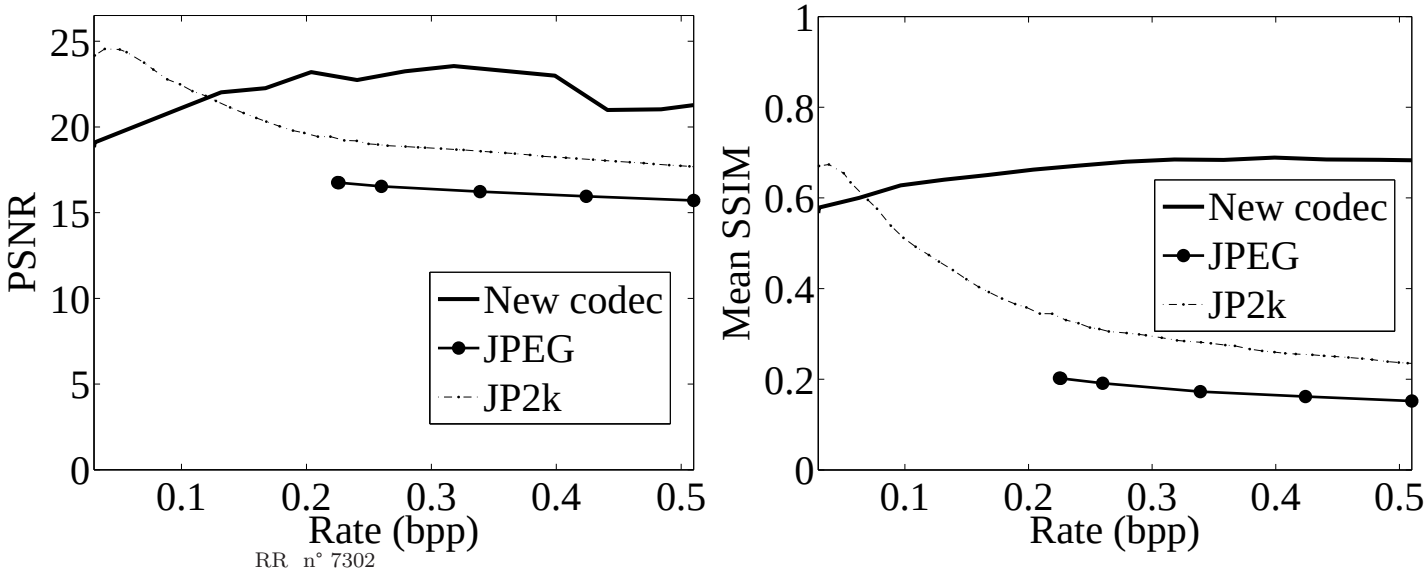


Figure 23: Rate/Quality behavior: quantitative comparison between our new codec, JPEG, and JPEG2000 using two different quality measures.

6 Discussion

We have proposed a new bio-inspired codec for static images. First, the image is converted into a ROC code via a simplified retinal model, then a stack run coder is applied, followed by a first order arithmetic compressor. The performances of this coding scheme were tested against well established JPEG standards, and we obtain encouraging results for low bandwidth transmissions, especially when dealing with noisy data. This compression scheme also offers interesting features such as scalability and reasonable complexity. Limitations have been observed in terms of rate/quality, when compared to JPEG2000 for noiseless data transmissions. Beyond the proposition of a new compression scheme, we would like to highlight a variety of important issues and present potential avenues for future research efforts in this direction.

The first perspective concerns the retina coding model of our scheme. Although we focused on the latency time of the first spike, several models take into account the whole structure of spike trains. For example, it appears that burst or synchronies are features that could encode for the stimulus. This opens new perspectives to extend this model as soon as we are able to produce realistic spike trains. In particular, we will need to consider more realistic retina models converting videos into spike trains, such as [WK09]. The goal is then to use such models in order to reproduce some spiking pattern as observed in real cell recordings, and establish how spikes are triggered by a stimulus then decoded by the nervous system [RWdRvSB97].

The second perspective concerns compressing spikes. In this paper, even with a simplified representation of the spiking activity as a wave of spikes, classical approaches as stack run coding are not optimal. In the general case, with a continuous and intense spiking activity, new ideas will have to be introduced. New bio-inspired compression schemes will have to take into account the features of the neural code that are the most relevant for the stimulus representation.

References

[Adr26] E.D. Adrian. The impulses produced by sensory nerve endings. *Journal of Physiology*, 61(1):49–72, March 1926.

[BCK07] E. Bodden, M. Clasen, and J. Kneis. Arithmetic coding revealed: A guided tour from theory to praxis. Technical report, McGill University School of Computer Science Sable Research Group, 2007.

[BMLF] P. Baudot, O. Marre, M. Levy, and Y. Frgnac. Nature is the code: High reproducibility in v1 and a model of optimal dissipation of visual structural complexity. (to be submitted).

[BRdRvSW91] W. Bialek, F. Rieke, R. de Ruyter van Steveninck, and D. Warland. Reading a neural code. *Science*, 252(5014):1854–1857, June 1991.

[BWC90] T.C. Bell, I. H. Witten, and J.G. Cleary. *Text compression*. Prentice Hall, Englewood Cliffs, N.J. :, 1990.

[CCG+06] C.G.S. Cardoso, M.C. Cunha, A. Gomide, D.J. Schiozer, and J. Stolfi. Finite elements on dyadic grids with applications. *Mathematics and Computers in Simulation*, 73(1–4):87–104, November 2006. Applied and Computational Mathematics - Selected Papers of the Fifth PanAmerican Workshop - June 21-25, 2004, Tegucigalpa, Honduras.

[CPK+07] J. Cho, A.R.C. Paiva, S. Kim, J.C. Sanchez, and J.C. Príncipe. Self-organizing maps with dynamic learning for signal reconstruction. *Neural Networks*, 20(2):274–284, March 2007.

[CSE00] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG2000 still image coding system: An overview. *IEEE Transactions on Consumer Electronics*, 16(4):1103–1127, 2000.

[CT91] T.M. Cover and J.A. Thomas. *Elements of information Theory*. Wiley-Interscience, August 1991.

[dRvSB88] R. de Ruyter van Steveninck and W. Bialek. Real-time performance of a movement-sensitive neuron in the blowfly visual system: Coding and information transfer in short spike sequences. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 234(1277):379–414, September 1988.

[Fie94] D.J. Field. What is the goal of sensory coding? *Neural Computation*, 6(4):559–601, July 1994.

[GG92] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*, volume 159 of *The Springer International Series in Engineering and Computer Science*. 1992.

[Gir93] B. Girod. What's wrong with mean-squared error?, digital images and human vision. *Cambridge, MA, MIT Press*, pages 207–220, 1993.

[GM08] T. Gollisch and M. Meister. Rapid neural coding in the retina with relative spike latencies. *Science*, 319(5866):1108–1111, February 2008.

[GT98] J. Gautrais and S. Thorpe. Rate coding vs temporal order coding : a theoretical approach. *Biosystems*, 48(1):57–65, 1998.

[Ham74] P. Hammond. Cat retinal ganglion cells: size and shape of receptive field centres. *The Journal of Physiology*, 242(1):99–118, October 1974.

[HR08] Niall Hurley and Scott Rickard. Comparing measures of sparsity (submitted). *submitted to IEEE Transations on Information Theory*, November 2008.

[Jr.84] G. G. Langdon Jr. An introduction to arithmetic coding. *IBM J. RES. DEVELOP*, 1984.

[KC03] J. Karvanen and A. Cichocki. Measuring sparseness of noisy signals. In *Proceedings of 4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pages 125–130, Kyoto, Japan, April 2003. Riken, ICA.

[Koh98] T. Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, November 1998.

[LKY98] B.B. Lee, J. Kremers, and T. Yeh. Receptive fields of primate retinal ganglion cells studied with a novel technique. *Visual Neuroscience*, 15(1):161–175, January–February 1998.

[LS04] N.A. Lesica and G.B. Stanley. Encoding of natural scene movies by tonic and burst spikes in the lateral geniculate nucleus. *Journal of Neuroscience*, 24(47):10731–10740, November 2004.

[LS07] S. Lyu and E.P. Simoncelli. Statistically and perceptually motivated nonlinear image representation. In B Rogowitz, T N Pappas, and S J Daly, editors, *Procedings SPIE, Conference on Human Vision and Electronic Imaging XII*, volume 6492, San Jose, CA, USA, January 2007. Society of Photo-Optical Instrumentation.

[Mal89] S.G. Mallat. A theory for multiresolution signal decomposition : the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.

[MI99] M. Meister and M.J. Berry II. The neural code of the retina. *Neuron*, 22(3):435–450, March 1999.

[NTACB07] S. Narasimhan, M. Tabib-Azar, H.J. Chiel, and S. Bhunia. Neural data compression with wavelet transform: A vocabulary based approach. In *Proceedings of the 3rd International IEEE EMBS Conference on Neural Engineering*, pages 666–669, Kohala Coast, Hawaii, USA, May 2007.

[OF96] B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive-field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

[OP92] M.W. Oram and D.I. Perrett. Time course of neural responses discriminating different views of the face and head. *Journal of Neurophysiology*, 68(1):70–84, July 1992.

[OSL01] B.A. Olshausen, P. Sallee, and M.S. Lewicki. Learning sparse image codes using a wavelet pyramid architecture. In *Advances in Neural Information Processing Systems*, volume 13, pages 887–893. MIT Press, 2001.

[PB68] D. Perkell and T. Bullock. Neural coding. *Neurosciences Research Program Bulletin*, 6:221–343, 1968.

[Per03] L. Perrinet. *Comment dchiffrer le code impulsionnel de la Vision? tude du flux parallle, asynchrone et pars dans le traitement visuel ultra-rapide*. PhD thesis, Universit Paul SABATIER, February 2003.

[PPS05] A.R.C. Paiva, J.C. Príncipe, and J.C. Sanchez. Compression of spike data using the self-organizing map. In *Proceedings of 2nd International IEEE EMBS Conference on Neural Engineering*, pages 233–236, Washington D.C., USA, March 2005.

[PST04] L. Perrinet, M. Samuelides, and S. Thorpe. Coding static natural images using spiking event times: do neurons cooperate? *IEEE Transactions on Neural Networks*, 15(5):1164–1175, September 2004.

[Rod65] R.W. Rodieck. Quantitative analysis of the cat retinal ganglion cells response to visual stimuli. *Vision Research*, 5(11):583–601, December 1965.

[RPP07] S. Rao, A.R.C. Paiva, and J.C. Príncipe. A novel weighted lbg algorithm for neural spike compression. In *Proceedings of International Joint Conference on Neural Networks*, pages 1883–1887, Orlando, FL, USA, August 2007.

[RT01] R. Van Rullen and S. Thorpe. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural computation*, 13(6):1255–1283, June 2001.

[RT02] R. Van Rullen and S. Thorpe. Surfing a spike wave down the ventral stream. *Vision Research*, 42(23):2593–2615, October 2002.

[RWdRvSB97] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek. *Spikes: Exploring the Neural Code*. The MIT Press, Cambridge, MA, USA, 1997.

[Say00] K. Sayood. *Introduction to Data Compression, Second Edition (Morgan Kaufmann Series in Multimedia Information and Systems)*. Morgan Kaufmann, 2000.

[SCE01] A. Skodras, C. Christopoulos, and T. Ebrahimi. JPEG2000: The upcoming still image compression standard. *Pattern Recognition Letters*, 22(12):1337–1345, 2001.

[SF95] E.P. Simoncelli and W.T. Freeman. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *Proceedings of International Conference on Image Processing*, volume 3, pages 444–447, Washington, DC, USA, October 1995.

[SF07] B. Sen and S. Furber. Maximising information recovery from rank-order codes. In *Proceedings of SPIE conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, volume 6570, page 65700C, Orlando, FL, U.S.A., April 2007.

[SLD99] G.B. Stanley, F.F. Li, and Y. Dan. Reconstruction of natural scenes from ensemble responses in the lateral geniculate nucleus. *Journal of Neuroscience*, 19(18):8036–8042, September 1999.

[TFM96] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, June 1996.

[Tho90] S. Thorpe. Spike arrival times: A highly efficient coding scheme for neural networks. *Parallel processing in neural systems and computers*, pages 91–94, 1990.

[TVC96] M.J. Tsai, J.D. Villasenor, and F. Chen. Stack-run image coding. *IEEE transactions on circuits and systems for video technology*, 6(5):519–521, October 1996.

[Vic87] J.D. Victor. The dynamics of the cat x cell center. *Journal of physiology*, 386:219–246, May 1987.

[VSN03] R. Valerio, E.P. Simoncelli, and R. Navarro. Directly invertible nonlinear divisive normalization pyramid for image representation. *Visual Content Processing and Representation*, 2849:331–340, September 2003.

[VT01] R. VanRullen and S. Thorpe. Is it a bird? is it a plane? ultra-rapid visual categorisation of natural and artifactual objects. *Perception*, 30(6):655–668, 2001.

[WBSS04] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[WK09] A. Wohrer and P. Kornprobst. Virtual retina : A biological retina model and simulator, with contrast gain control. *Journal of Computational Neuroscience*, 26(2):219–249, 2009.

[WKA09] Adrien Wohrer, Pierre Kornprobst, and Marc Antonini. Retinal filtering and image reconstruction. Research Report RR-6960, INRIA, 2009.

[Woh07] A. Wohrer. Mathematical study of a neural gain control mechanism. Research Report RR-6327, INRIA, 2007.

[Woh08] A. Wohrer. *Model and large-scale simulator of a biological retina, with contrast gain control*. PhD thesis, Graduate School of Information and Communication Sciences, University of Nice-Sophia Antipolis, 2008.

[WW02] W.Gerstner and W.Kistler. *Spiking Neuron Models : Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

Contents

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399